REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching e
the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions
Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Re

AFRL-SR-BL-TR-00-
0637

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYP
	1 Nov 00	Final Tech Report 01 Dec 97 to 31 Dec 99

Develop

4. TITLE AND SUBTITLE
~~Devlop~~ Equalization Techniques for use with Bandwidth Efficient Modulatiuon Methods

5. FUNDING NUMBERS
F49620-98-1-0102

6. AUTHOR(S)
Sheila Horan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
New Mexico State University
College of Engineering
P. O. Boc 30001
Los Cruces, NM 88003-8001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
AFOSR/NM
801 N. Randolph Street, Rm 732
Arlington, VA 22203-1977

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-98-1-0102

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Bandwidth is a precious commodity. In order to make the best use of what is available, better modulation schemes need to be developed, or less data needs to be sent. This paper will investigate the option of sending less data via data compression. The structure and the entropy of the data determine how much lossless compression can be obtained for a given set of data. This paper shows the data structure and entropy for several actual telemetry data sets and the resulting lossless compression obtainable using data compression techniques.

DTIC QUALITY INSPECTED 4

20001205 078

14. SUBJECT TERMS	15. NUMBER OF PAGES
	160
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NOV 0 1 2000

Final Report

Develop Equalization Techniques for use with Bandwidth Efficient Modulation Methods

Control Number 97NM291
USAF Grant Number F49620-98-1-0102

August 2000

Sheila Horan
College Associate Professor

New Mexico State University

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

Table of Contents

Abstract	3
Project Summary	4
Introduction	6
Test Data	9
Channel Error Simulations	14
Channel Coding	15
Conclusion	16
References	16
Acknowledgements	16
Appendix – Computer Programs	17
Appendix – Data Statistics	134

Final Report for Developing Equalization Techniques for use with Bandwidth Efficient Modulation Methods

Subtitle: Data Compression Techniques to Reduce Bandwidth

Abstract

Bandwidth is a precious commodity. In order to make the best use of what is available, better modulation schemes need to be developed, or less data needs to be sent. This paper will investigate the option of sending less data via data compression. The structure and the entropy of the data determine how much lossless compression can be obtained for a given set of data. This paper shows the data structure and entropy for several actual telemetry data sets and the resulting lossless compression obtainable using data compression techniques.

Final Report for Developing Equalization Techniques for use with Bandwidth Efficient Modulation Methods

Subtitle: Data Compression Techniques to Reduce Bandwidth

Project Summary:

Objectives:

1. Analyze the performance of the 32 QAM in multipath channels.
2. Develop adaptive filtering equalization for 32 QAM in multipath channels.
3. Analyze the performance of pulse shaped 8 PSK in multipath channels.
4. Develop adaptive filtering equalization for 8 PSK in multipath channels.
5. Investigate the use of interleaving for the 32 QAM and 8 PSK modulation techniques in a multipath environment.
6. Develop reports detailing work and recommendations.
7. Perform statistical analysis on the telemetry data to determine possible amounts of compression.
8. Apply arithmetic, Ziv-Lempel, Huffman, and Rice data compression techniques to the data.
9. Test data compression techniques in a channel simulation to test the robustness of the code.
10. Adapt the Rice algorithm to better handle telemetry data.
11. Complete Final report.

Accomplishments:

Objectives 1 through 6 have not been met and were changed. To test the 32 QAM and 8 PSK in the multipath channels required the appropriate channel model be available. Brigham Young University (BYU) was developing this model when this project started, and the model was not available. During the progress of this project, the objectives changed. Discussion with ARTM led to a different area of concentration. Work by others was proving successful with FQPSK, so this work on modulation was changed to an emphasis on data compression. Reducing the amount of data to be sent can reduce the bandwidth and thus accomplish the same results as the 8 PSK, and 32 QAM.

Objectives 7 through 9 were completed. Data compression results indicate that data compression is viable, channel simulations do indeed indicate that data compression alone would be unfeasible, but adding channel coding to protect the data again makes this data compression with channel coding a viable option.

Objective 10 is still in process. The work was not completed due to the extensive amount of time it would take to complete the task, and student employment problems. Current results do indicate that an adaptation of the Rice algorithm would work well for telemetry data.

Personnel Support:

1. One Faculty member (the PI)
2. One technician (for help in setting up the computers in the lab)
3. Five undergraduate students (total at various times, no student worked more than part-time. Two students worked for 1year, and 2 other students worked for 6 months. The last student was supported by an on campus research grant for 2.5 months) The students were:
   Jonathan McMillan – worked on learning simulations and getting simulink to work; attended 1998 ITC conference.
   James Woods – wrote programs for statistical analysis, attended 1998 ITC conference.
   Shakti Davis – worked on the error analysis, and channel simulations, attended 1999 ITC conference
   Kendall Mauldin – began looking into the Rice algorithm and made recommendations to modify it, attended 1999 ITC conference.
   Enrique Sanchez – started modifications of Rice algorithm but was unable to complete the work.

Publications:

Paper at the International Telemetering Conference:
"Data Compression Statistics and Implications", ITC October 1999

Interactions/Transitions:

1. Participated in Advanced Range Telemetry (ARTM) group meeting in the spring (1998).
2. Consultative/advisory functions: Communicate with BYU regarding our related projects
3. Participated in Advanced Range Telemetry (ARTM) group meeting at the ITC (1998, 1999)
4. Participated in Advanced Range Telemetry (ARTM) communications via email and phone. New error sets were to be sent in January (1999) – none received as of this date.

New Discoveries, inventions, or patent disclosures: None

Honors/Awards: None

Introduction

Bandwidth is a precious commodity. It has become imperative that bandwidth is used efficiently. To this end, data needs to be reduced, or modulation techniques need to be used t minimize the bandwidth. Data compression (coding) is one technique to reduce the bandwidth needed to transmit data. The bit rate = R = (# of data bits sent)/ sec. The bandwidths for BPSK are (6):

3-db bandwidth	0.88 R
null to null bandwidth	1.0 R

As can be seen if R decreases, the so to does the bandwidth. The goals of data coding are to:

- Reduce the amount of data
- Find codes that take into account redundancy, structure, and patterns
- Break up data into random groups

Entropy is a measure of the information content of the data, a measure of the uncertainty or randomness of the data, and can be used to indicate how many bits would be needed to encode the data if the data are independent.
If X is a random variable, process, or data to be coded (usually X is discrete, but doesn't have to be); which takes on the values $X_1, X_2, \ldots, X_n$; and p_i is the probability that X_i occurs, then the entropy

$$H(X) = -\sum_{\text{all } i} p_i \log(p_i)$$

If the log is base two, then the units will be bits and will indicate the number of bits needed to code the message without loss of information. In order to calculate this quantity, the all of the marginal and joint probabilities have to be known. This is not usually possible. Most of the time, the entropy is estimated by the first order probabilities (that is the probabilities of the individual symbols). A second order probability is based on grouping two symbols at a time; third order is three symbols at a time. etc. The true entropy is based on the nth order probabilities as n approaches the size of the data file to be transmitted.
Independence of data means that the probability of (X_1X_2) is the product of the individual probabilities.

Example:

Suppose that the original data is given by:
0 1 2 2 2 3 4 4 4 5 6 6 7 8 8 8

To code this directly in binary would take 4 bits per symbol. The amount of data compression can be found by using entropy and assuming independence of the data. To find the first order entropy, the probability of each symbol would need to be found.

Symbol	Freq.	p_i = freq/(total # symbols)	$-p_i \log p_i$
0	1	0.0625	0.25
1	1	0.0625	0.25
2	3	0.1875	0.453
3	1	0.0625	0.25
4	3	0.1875	0.453
5	1	0.0625	0.25
6	2	0.125	0.375
7	1	0.0625	0.25
8	3	0.1875	0.453
overall	16	1.0	Data entropy=2.983

This gives a reduction of about 25%. The % compression can be found by

$$\text{compression ratio} = 1 - \frac{\text{compressed file size (average coded symbol size)}}{\text{orignal file size (uncoded symbol size)}}$$

If the data are independent and if the above is an accurate representation of the entropy, then the best compression that would be possible is 2.983 bits/symbol.
The following table shows the results obtained if the data are grouped by two symbols at a time:

Two bits	New symbol	Freq	Prob	$-P_i \log P_i$
01	0	1	0.125	0.375
22	1	1	0.125	0.375
23	2	1	0.125	0.375
44	3	1	0.125	0.375
45	4	1	0.125	0.375
66	5	1	0.125	0.375
78	6	1	0.125	0.375
88	7	1	0.125	0.375
totals		8	1.0	3.0

This gives an entropy of 3.0 (still a decrease from the original, but an apparent increase from the first order entropy). This really isn't an increase. This entropy is for two symbols at a time and the compression ratio will be 1-(3x8)/(4x16) = 63%.
These entropies were calculated assuming independence of the data. Suppose there is a relationship between the data. One way to examine this is to look at the differences between neighboring data. To start, leave the first data symbol alone, and then subtract the second from the first, then the third from the second, etc. This gives:
Original data: 0 1 2 2 2 3 4 4 4 5 6 6 7 8 8 8
Differenced: 0 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0

In this case, there are two possible symbols (0 and 1), each occurring 8 times. This will give a probability of ½ for each symbol, and an entropy of 1. We could now code the original data with 16 bits of data where the original would take 4 times that many bits. This would give a compression ratio of 75% over the original data. Since we get compression by differencing the data, this indicates that there is correlation between the data and that they are not independent. Suppose now that the differenced data are grouped two at a time as follows:

01 10 01 10 01 10 11 00

Analysis of this data gives

Two bits	New symbol	Freq	Prob	$-P_i \log P_i$
01	0	3	0.375	0.531
10	1	3	0.375	0.531
11	2	1	0.125	0.375
00	3	1	0.125	0.375
totals		8	1.0	1.811

The compression ratio for this grouping is: 1- (8x1.811)/(4x16) = 77%
Notice how the compression ratio continues to increase, meaning that the file size gets smaller. Continuing to group one more time – grouping by fours gives:

0110 0110 0110 1100

This has just two symbols. 0110 occurs three times, and 1100 occurs only once. This will give:

Two bits	New symbol	Freq	Prob	$-P_i \log P_i$
0110	0	3	0.75	0.311
1100	1	1	0.25	0.5
totals		4	1.0	0.811

In this case, with just two symbols to code, the compression ratio becomes:

1- (4x0.811)/(4x16) = 95%

The above is just an example to demonstrate how compression can occur. The compression ratio is just a prediction of the actual compression. In order to achieve this predicted compression, an appropriate compression algorithm would need to be developed. But this can act as a guide for how well the compression could do. If all of the probabilities were known for the data, then entropy would give a lower bound on the amount of compression to be expected. As higher order probabilities are used, the predicted compression will become closer to the entropy bound.
The above ideas of data compression and entropy will now be applied to Telemetry data.

Test Data

Thirteen actual telemetry data files were obtained from Vern Diekman at TYBRIN (1) through the Advanced Range Telemetry (ARTM) group. The data were then analyzed to find the 4 bit, 8 bit, and 12 bit entropies. The entropy for a set of data is given by:

$$H(S) = -\lim_{n\to\infty}\frac{1}{n}\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m}\{P(X_1=i_1,X_2=i_2,\ldots,X_n=i_n)\bullet \log P(X_1=i_1,X_2=i_2,\ldots,X_n=i_n)\}$$

(1)

Where H (S) is the entropy of the source (the data)

X_i are the elements of all possible sequences of the data. The X_i can be bits, symbols, or words.

n = the length of the sequence

P is the probability

m = the size of the alphabet used for coding; for binary data, m=2, for words of 4 bits in length,

m=16, etc.

To find the 4-bit entropy, all possible combinations of four elements from the binary alphabet were found. The frequency of occurrence of each of these 4 bit words was found. This frequency was used as an estimate of the probability of occurrence for each of these words. From these probabilities, the estimate for the 4 bit entropy was found using equation 1 where n=4 and m=2. For the 8-bit entropy, 256 different words are possible, and for the 12-bit entropy, 4096 words are possible. It can be seen that letting n approach infinity will quickly become impractical. The formats for each data set varied. Some of the data were coded into words of 10 bits, some 12 bits, etc. Plots and statistics for each of the data sets are in the appendix. A summary of the data statistics is given below:

Data Set Entropies and Word Size					
Data Set	Data word size In bits	4 bit Entropy	8 bit Entropy	12 bit Entropy	#bits/file
SDS001	12	3.9	7.5	10.3	139536000
SDS002	12	3.3	6.0	7.5	94239360
SDS003	12	3.8	7.0	9.0	157352160
SDS004	10	3.4	5.9	7.3	496743000
SDS005	10	3.6	6.6	8.6	1234710000
SDS006	10	3.7	5.9	7.6	890556000
SDS007	10	2.9	5.9	7.5	2347503000
SDS008	12	2.6	4.2	4.7	1105437696

Data Set Entropies and Word Size					
Data Set	Data word size In bits	4 bit Entropy	8 bit Entropy	12 bit Entropy	#bits/file
SDS009	12	3.2	5.8	6.9	1102886400
SDS010	12	2.7	4.3	4.8	1100703744
SDS011	12	2.8	4.9	5.9	568813056
SDS012	16	2.5	3.7	5.2	2193232356
SDS013	24	2.8	3.7	4.6	3443281920

To determine the amount of compression that should be possible, the entropy per number of bits must be used. Hence % compression for k bit entropies = (1-(k bit entropy / k))*100. The amount of possible compression for each data set can then be found. If the k bit entropy is equal to the entropy, then the predicted compression will be a lower bound for all data compression techniques. A plot of the predicted compression from the calculated entropies is given in Figure 1. It can be seen that the compression increases as the number of bits per symbol is increased. This indicates that the entropies do not yet equal the entropy for the data sets. Consequently, compression techniques should be able to achieve values better than these predicted values.

The Huffman, an Adaptive Huffman, Arithmetic, and Lempel-Ziv compression algorithms were applied using programs from Mark Nelson's text [2]. The Huffman codes and Arithmetic codes use an 8-bit word length in the code. Two variations of the Lempel-Ziv algorithm were used. The LZSS uses a pair of values to indicate the location of the match and the length of the match in the dictionary. The LZW algorithm involves only sending one element instead of a pair of elements, and using a start up alphabet in the dictionary consisting of all the letters of the source alphabet. The percent compression for each technique along with the predicted compression is given in the following table.

Data Compression Results								
Data Set	Huffman	Adapt. Huffman	LZSS	LZW	Arith-metic	% 4 bit Predicted compress	% 8 bit Predicted compress	% 12 bit predicted compress
SDS001	6.3	6.9	15.5	-9.9	6.6	2.5	6.3	14.2
SDS002	24.1	25.5	61.1	33.8	24.7	17.5	25.0	37.5
SDS003	11.5	15.5	39.7	-7.3	11.9	5.0	12.5	25.0
SDS004	25.1	27.7	50.2	11.5	25.5	15.0	26.3	39.2
SDS005	18.5	19.0	34.7	18.5	18.7	10.0	17.5	28.3
SDS006	25.1	25.9	40.2	32.5	25.4	7.5	26.3	36.7
SDS007	25.9	26.5	41.9	28.2	41.9	27.5	26.3	37.5
SDS008	46	48.3	71.0	53.9	46.7	35.0	47.5	60.8
SDS009	27.5	29.4	64.7	26.3	27.7	20.0	27.5	42.5
SDS010	45.2	47.5	68.9	52.7	45.9	32.5	46.3	60.0
SDS011	37.6	39.7	64.5	46.0	38.1	30.0	38.8	50.8
SDS012	51.9	51.9	65.9	63.7	51.9	37.5	53.8	56.7
SDS013	48.6	53.7	65.5	64.8	51.4	30.0	53.8	61.7

A negative with the compression value indicates that the file was expanded instead of compressed. It can be observed that each data file compressed differently. Figure 2 contains the plot of the results of the compression techniques. It can be seen from Figure 2 that certain files will compress very well. A 60% compression would mean that the file would take up less than half its original size. In all cases, there is at least one technique that provides compression of 10% or more. With the demands on spectra, even this little gain can be worth something. Since the Huffman and Arithmetic codes that were tried work with 8 bit word sizes, if the compression obtained by these techniques is compared with the 8-bit entropy, we see a very close match. All but 2 of the results are within 10% or less of the 8 bit entropy. Since the LZ algorithms achieve larger compression than this indicates, the 8-bit word size is not the best choice and that this 8-bit entropy is not the true entropy. Also, since the k bit entropies continue to increase with k, this also indicates that the actual entropy has yet to be found. Hence even larger compressions can be expected.

The Rice data compression algorithm was also tried. The results were discouraging. Instead of any compression, files were either expanded or were unable to decode to regain the original file. The main problem with the Rice algorithm is that is was designed specifically for images. The next thing tried was to modify the Rice code written by UNM to run with variable word length, and variable number of words per frame. This was found to be much more work because of the following problems:

1. The code is written and optimized to deal with image files, not telemetry data. This causes several "domino-effect" problems.
2. A "long" (32-bit variable) is used to keep track of the number of pixels (bits in our case) in a file. The largest value possible for a "long" is 2,147,483,647. So if a file size ever exceeds 2,147,483,647 bits (268,435,456 bytes = approx. 256 Megabytes), we would hit this limitation – one telemetry file exceeds this limit.
3. ARTM telemetry is organized in frames with length of 200 words each. The UNM code (without modification) doesn't support this frame size or structure.
4. As said in a comment on line 3734 of rice.c (UNM code) the hardware (and thus the C code that simulates the chip) is limited to 64 blocks per scanline.
5. There also seems to be a magic number of 64 when it comes to the "pixels per block" command-line option of the compression program. When this option (-j <integer number here>) is set to 64 or less, there is no problem compressing or UNCOMPRESSING the file. When it is set to >64 there are no problems compressing the file, by when the file is UNCOMPRESSED, the uncompression part of the program gives an error message and halts. Also, the UNM code truncates the data frame if it is at the end of the file and its length is less than a complete scan line.

Noting these problems, the task of re-writing the code was undertaken, but time ran out before the task was completed. In an effort to decide whether this work would be worth the effort, other work was investigated.

Figure 1. Predicted Compression from Data Statistics

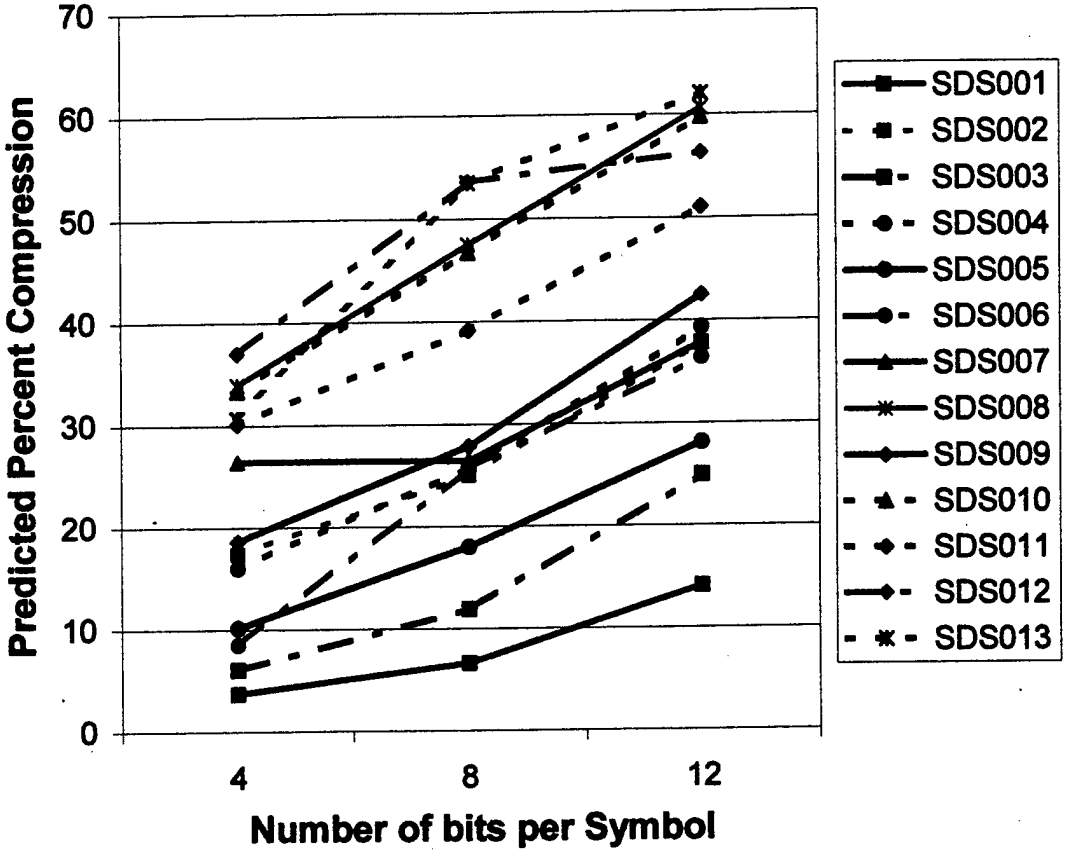
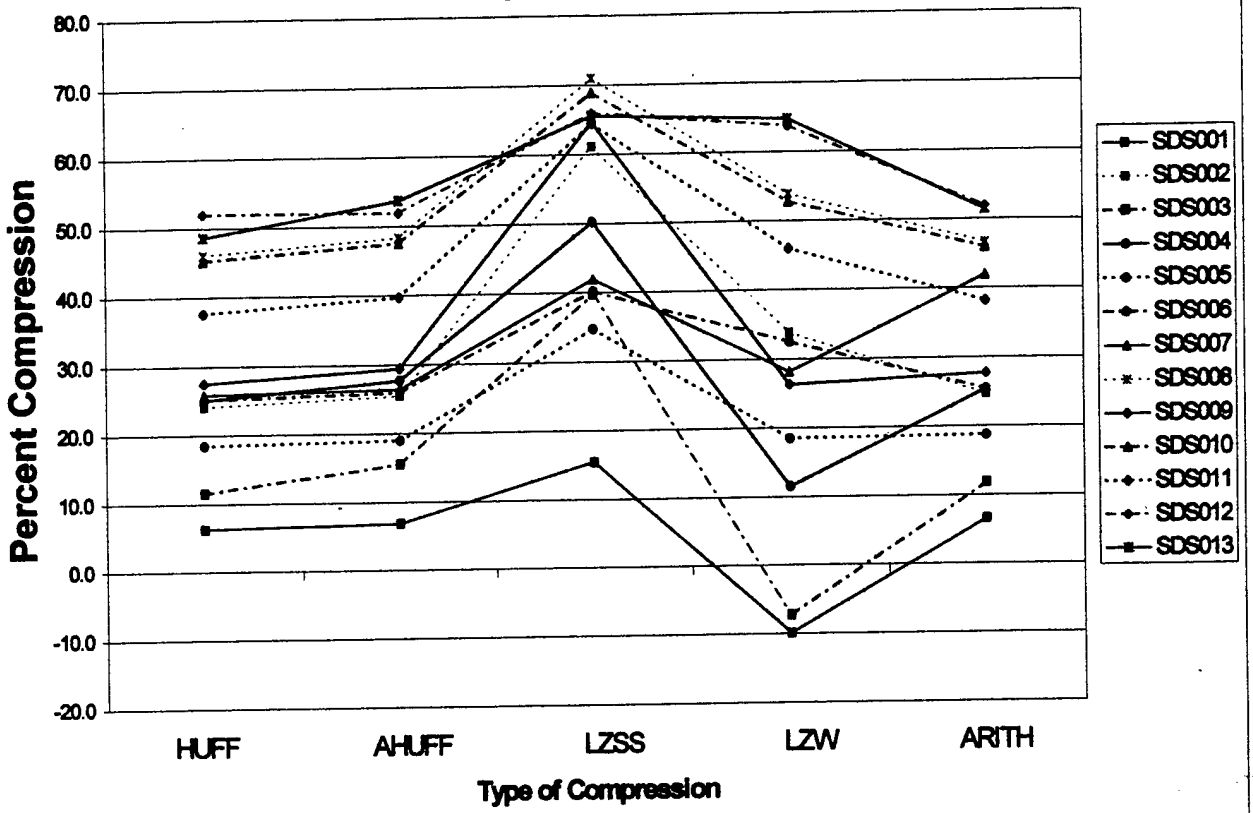


Figure 2.
Data Compression Results



In data compression classes at NMSU, students have applied various compression techniques to data. One student even added errors to see the effect errors would have on a compressed file. In an effort to see how well the compression could work, the results for an image is summarized below (3):

Compression Algorithm	% Compression	Channel Error rate	Errors made	Comments
None	0%	$1x10^{-6}$	1	Can't notice error
None	0%	$1x10^{-4}$	58	Slight speckling
None	0%	$1x10^{-2}$	5022	Lots of speckles
Huffman	12%	$1x10^{-6}$	1	Shift in image
Huffman	12%	$1x10^{-4}$	42	Distorted, but recognizable
Huffman	12%	$1x10^{-2}$	4562	Decoder failed
Ziv-Lempel	3%	$1x10^{-6}$	1	Can't notice error
Ziv-Lempel	3%	$1x10^{-4}$	58	Highly distorted
Ziv-Lempel	3%	$1x10^{-2}$	4981	Decoder failed
Arithmetic	13%	$1x10^{-6}$	1	Decoder failed
Arithmetic	13%	$1x10^{-4}$	58	Decoder failed
Arithmetic	13%	$1x10^{-2}$	4505	Decoder failed
Rice	22%	$1x10^{-6}$	1	Decoder failed
Rice	22%	$1x10^{-4}$	5	Decoder failed
Rice	22%	$1x10^{-2}$	32	Decoder failed

From this table, it is obvious that single errors with compressed data can be disastrous. It can also be seen that when the Rice algorithm is used for images (for which it was designed), the compression is almost doubled. If the Rice algorithm can be adapted to handle 10, 12, 16 and 24 bits per word, and handle variable word sizes per frame (40, 128, 160, 200, 256, 300 etc) then the correlations of the data could be used to help reduce the file size. Since the data from one frame to another would be similar, taking advantage of these similarities should aid in the compression of the data. One way the Rice algorithm uses to reduce data correlation is to subtract the following symbol from the previous one. This works well for images, but not telemetry. In telemetry an entire frame would need to be subtracted from the following frames to reduce the correlation. The Rice algorithm is not currently set up to handle the variable frame sizes needed.

Channel Error Simulations

Error Sample File Description

Error files were obtained from Bert Carner at Synthesys Research (4) through ARTM for channel error simulations.

General Conditions:

Frequency: L band

Purpose of test: A/B comparison of FM and FQPSK-B @ 1 Mb/s
TX power: nine Watts, each transmitter
TX antenna pattern: well behaved omni patterns on belly of aircraft, single antenna
RX antenna: 8-foot dish, bldg 5790
Receivers: Microdyne model 700, narrowband, IF BW 1 MHz, single channel, i.e., NO COMBINING.

Sample point conditions:

Straight and level flight, Black Mountain corridor, 5k foot pressure altitude, approximately 2500 foot AGL, 200 Kt indicated airspeed

The two points represent classic multipath events. In this case there are at least three rays. There is a strong short delay ray (10-20 nsec range) and a weak long delay influence (in the range of 2-5 usec)

The main event in each file lasts about 1.5-2 seconds and the null depth is deep, at least 30 dB. The slant range is only about 40 miles, very strong signal conditions (Eb/No greater than 20 dB).

Two representative files were used to demonstrate the effect of the errors on the compressed files. For the data files SDS008, and SDS007 we have the following results:

BERA Simulation Results

File name	Compressed?	Num Errors	Num Bits	BER
Sds008.dat	No	3	1105437696	2.713857e-009
Sds008.dat	Yes	2212390	8650752	0.2557
Sds007.dat	No	10	2.3475e+009	4.2599e-009
Sds007.dat	Yes	218	2.3475e+009	9.2865e-008

The compression algorithm used for compression on the above was the Ziv-Lempel algorithm (since it tends to give the best compression results). Here, the errors can cause major problems. Why does it create so many more errors in the compressed case? That is based on how the individual methods of compression work. With Ziv-Lempel (a dictionary based scheme), an error in the data, means the error will appear in the dictionary and so affect other code words. So, in the case of SDS008, the original three errors could expand to 2212390 errors as the errors go through the dictionary formed by the compression algorithm. However, if channel coding is used, these problems can be avoided. Other ways to prevent the propagation of errors is to reset the code after so many bits and restart the code.

Channel Coding

Looking at the SDS008 case where 3 errors occur in the original file, channel coding would be able to eliminate any errors. A BCH code (1023, 973) can correct 5 errors (5). The 973 is the number of incoming bits to be channel coded and the 1023 is the number of bits sent. For each 973 bits input into the coder, the coder will add 50 bits so that it can find up to and including 5 errors per 1023 bits received. For this file, 8891 blocks

will need to be channel coded, giving a total of 9,095,302 bits to be added back into the file to eliminate the errors caused by the channel. Still at this rate, the amount of compression would be:

Compression ratio = {1- (8,650,752+9,095,302)/(1,105,437,696) } x100 = 98%

This still gives a sizeable amount of compression, and ensures protection from the channel. There are many different types of channel noise and each can treated using different techniques. BYU is currently working of ways of handling the channel effects that are pertinent to the problems encountered by the military.

Conclusions

Data compression is a viable option to aid in reducing or maintaining bandwidth. Even with channel errors figured in, the compression achieved is worth the effort involved. Significant compression can be achieved through off the shelf technology. Even more compression should be obtainable with modifications of the Rice algorithm.

Acknowledgements:

The author greatly appreciates the help received from the students who participated in this project and the people at ARTM (Chuck Irving and Vincent Schiappi) for their help and encouragement, Warner Miller at NASA for his help in obtaining the Rice coding, Gary Maki (University of New Mexico) for helping with the Rice coding, AFOSR for their support of this project, and especially Vern Diekman at TYBRIN for his help with the data sets.

REFERENCES:

[1] Vern Diekman, TYBRIN for Data sets

[2] Nelson, M., The Data Compression Book, M&T Books, California, 1987

[3] Wolcott, Ted final project report for data compression class, May 1994

[4] Carner, Bert, SynthesSys Research, error files, (650) 364-1853 or Bert_Carner@synthesysresearch.com

[5] Rice, M., ITC Short Course on Channel Coding, ITC Conference, 1999.

[6] Couch, Leon, Digital and Analog Communication Systems, Table 2-4, page 115, MacMillan Pub., 1993.

Computer Programs Appendix

James Woods wrote a lot of C code to examine the statistics of the data files and lay the preliminary work for the compression algorithms. The data and the initial data reading programs were obtained from Vernon Diekmann, at TYBRIN a subcontractor to Edward's Air Force Base. Some of the programs written by Vern Diekmann were modified slightly and are included in the list of programs below. The compression algorithms used were from Mark Nelson's book, except for the Rice algorithm, which was obtained through NASA and the University of New Mexico. A listing of the programs written by James Woods and a brief description of what they do is given below:

COMPRICE=> This program allows you to compare a file that has been run through rice to another file.
COMP2 , COMP6 => This program just compares two files on your hard drive.
COMP3CD => Takes a file from the CD and compares it to one from the hard drive.
COUNT12 => Counts 12 words configurations from the hard drive.
FRAMAV => Finds the average frame, the most popular frame.
FRAMINU1=> Takes the first frame and minuses it from the other frames.
FRAMMINU=>Takes a frame that was saved in another file and minuses it from the data file.
IMPROVE=>Same thing as display but from the hard drive.
NEGATIVE=>Minuses the frames without negatives.
NEW_TWO=>Counts the different configurations of 4 and 8 bits from the hard drive.
READ1=>Will let you read any data file you want to put in.
TEN_GREA=>Counts the different configuration of 10 bits from the CD.
TWOONE=>Counts the different configurations of 4 and 8 bit from the hard drive.
TWOONECD=>Counts the different configurations of 4 and 8 from the CD.
WORD=>Counts the different configurations of 12 bits from the hard drive.
WORDCD=>Counts the different configurations of 12 bits from the CD.

The FRAM files were to see if using the Rice algorithm where the frames are subtracted would be viable. We ran into some limits of the students' programming abilities at this point and didn't successfully recreate the files we were working with – this was due to mapping the negative data values into the positive values. These programs should not be run without fixing the mapping problem.

The Two and Word programs were used to obtain the probability densities and then the entropies of the individual files.

The program files are on the following pages.

COMPRICE

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a, size1, size2, z;
unsigned int data, word, interval, rs, sfid;
unsigned  int i,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int  nfrms[numfiles], wow[lengfiln];
unsigned long int j=0,nf=0, fsp[numfiles], nbps[numfiles];
char response, excel[32],newfile[lengfiln], newfile1[lengfiln];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
                unsigned int b15: 2;
```

```
        unsigned int b16: 2;
    } ;
    struct lnibles{
        unsigned int n1 : 4;
        unsigned int n2 : 4;
        unsigned int n3 : 4;
        unsigned int n4 : 4;
        unsigned int n5 : 4;
        unsigned int n6 : 4;
        unsigned int n7 : 4;
        unsigned int n8 : 4;
    };
    struct lbytes{
        unsigned int n1 : 8;
        unsigned int n2 : 8;
        unsigned int n3 : 8;
        unsigned int n4 : 8;
    };
    union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in ;
        union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in1 ;
        union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in2 ;
        union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
```

```
        } in3 ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp, *comp;/*File pointers*/

void main()
{
        i = 0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);/*Closed cinput.dat*/

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                        newfile[0]='e';
```

```
                newfile[1]=':';
                z=0;
                 do
                 {
                        newfile[2+z]=filename[fi][z];/*This puts 'e:' in front of the file
name so it can read it off the cd*/
                        z++;
                 }while(z<=10);z=0;
          f=fopen(newfile,"rb");/* Open selected data file */
          printf("\nWhat file would you like to compare that to: ");
          scanf("%s",newfile1);
          comp=fopen(newfile1,"rb");/*Open compare file and rb mean read in binary*/
          printf("\nWhat is the frame size after rice: ");
          scanf("%d",&size1);
          printf("\nWhat is the actual frame size before rice: ");
          scanf("%d",&size2);
          printf("\n\nWORKING!!!!\n");/*Know if it is working*/

          while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
          {       nf = nf + 1;/*Frame counter*/


                for(j=0; j<nwpf[fi]*nbpw[fi]/8; j++)/*Looping through the frame*/
                        {
                                fread(in1.data,sizeof(int),1,comp);/*Read in data
from compared file and store it in in1.data*/

                                if(in1.data[0]!=in.data[j])/*Checking if the two files
match 8 bits at a time*/
                                {
                                        if(in1.data[1]!=in.data[j])
                                        {
                                                printf("\nERROR at frame %ld and
in %d\n",nf,j);
                                                printf("\nActual %d and New data
%d\n",in.data[j],in1.data[0]);
                                                printf("\nDo you want to return to
the menu (y or n): ");
                                                response=getchar();
                                                if(response=='y')
                                                {
                                                        goto again;
                                                }
                                        }
                                }
                        }
                fread(in2.data,(size1-size2),1,comp);
          }
        printf("\nDONE COMPARING %s AND %s\n\n",filename[fi],newfile);
```

```
        printf ("Successful EndOfFile reached %ld Minor Frames.\n\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);/*Close the data file*/
        fclose(comp);/*Close the compare file*/
        nf = 0;
        again:/*This is where that goto function goes to*/
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*This prints menu of the
files*/
        printf ("  Enter the file Number\n");
        printf ("  or <ctrl> c to end: \n");
        }

} /* End Main */
```

COMP2, COMP6

```
include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data, word, interval, rs, sfid;
unsigned  int i,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int  nfrms[numfiles], wow[lengfiln];
unsigned long int j=0,nf=0, fsp[numfiles], nbps[numfiles];
char response, excel[32],newfile[numfiles][lengfiln];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
                unsigned int b15: 2;
```

```
        unsigned int b16: 2;
    } ;
    struct lnibles{
        unsigned int n1 : 4;
        unsigned int n2 : 4;
        unsigned int n3 : 4;
        unsigned int n4 : 4;
        unsigned int n5 : 4;
        unsigned int n6 : 4;
        unsigned int n7 : 4;
        unsigned int n8 : 4;
    };
    struct lbytes{
        unsigned int n1 : 8;
        unsigned int n2 : 8;
        unsigned int n3 : 8;
        unsigned int n4 : 8;
    };
    union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in ;
        union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in1 ;
    union
    {
        struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
        struct lnibles   nible[4096];
        struct lbytes    byte[4096];
        unsigned long int data[4096];
    } out ;
FILE *f, *fin, *fo, *lp, *comp;/*File pointers*/

void main()
{
    i = 0;

    fin=fopen("cinput.dat","rb");  /* Open file and read data description information */
    for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",
```

```
        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);/*Closed cinput.dat*/

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                f=fopen(&filename[fi][0],"rb");/* Open selected data file */
                printf("\nWhat file would you like to compare that to: ");
                scanf("%s",newfile[1]);
                hello=getchar();/*Take up the return*/
                comp=fopen(&newfile[1][0],"rb");/*Open compare file and rb mean read in
binary*/
                response='f';
                printf("\nWORKING!!!!\n");/*Know if it is working*/

                while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
                {       nf = nf + 1;/*Frame counter*/

                        if (response == 'f' || response == 'F')/* Display each minor frame */
                        {
                                while (fread(in1.data,(nwpf[fi]*nbpw[fi]/8),1,comp))/*Read in
data from compared file and store it in in1.data*/
```

```
                        {
                                for(j=0; j<nwpf[fi]*nbpw[fi]/8; j++)/*Looping through the
frame*/
                                {
                                        if(in1.data[j]!=in.data[j])/*Checking if the two files
match 8 bits at a time*/
                                        {
                                                printf("\nERROR in frame %ld and in
%ld",nf,j);
        printf("\n%02x\t%02x\n",in.data[j],in1.data[j]);/*Print in hex*/
                                                printf("\nDo you want to return to menu
(Y,N)\n");/*If the is a error this will let you return to the begining*/
                                                hello=getchar();
                                                if(hello=='y'||hello=='Y')
                                                {
                                                        goto again;/*If you want to know
where this goes to look for again at the bottom*/
                                                }
                                        }
                                }break;/*This breaks the while loop for the compare file so
that it read in a new frame*/
                        }
                }
        }
        printf("\nDONE COMPARING %s AND %s\n\n",filename[fi],&newfile[1][0]);
        printf ("Successful EndOfFile reached %ld Minor Frames.\n\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);/*Close the data file*/
        fclose(comp);/*Close the compare file*/
        nf = 0;
        again:/*This is where that goto function goes to*/
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*This prints menu of the
files*/
        printf ("  Enter the file Number\n");
        printf ("  or <ctrl> c to end: \n");
        }

} /* End Main */


#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
```

```
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data, word, interval, rs, sfid;
unsigned  int z=0,i,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int  nfrms[numfiles], wow[lengfiln];
unsigned long int j=0,nf=0, fsp[numfiles], nbps[numfiles];
char response, excel[32],newfile[lengfiln],file[lengfiln];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
                unsigned int b15: 2;
                unsigned int b16: 2;
        } ;
        struct lnibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
                unsigned int n3 : 4;
                unsigned int n4 : 4;
                unsigned int n5 : 4;
                unsigned int n6 : 4;
```

```
                unsigned int n7 : 4;
                unsigned int n8 : 4;
        };
        struct lbytes{
                unsigned int n1 : 8;
                unsigned int n2 : 8;
                unsigned int n3 : 8;
                unsigned int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in ;
                union
        {
                struct tbits   .tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in1 ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp, *comp;/*File pointers*/

void main()
{
        i = 0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
```

```
        }
        numfile = i ;
        fclose (fin);/*Closed cinput.dat*/

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                printf("\nWhat is the first file you want to compare: ");
                scanf("%s",newfile);
                f=fopen(newfile,"rb");/* Open selected data file */
                printf("\nType in the path and file name you would like to compare that to:\n");
                scanf("%s",file);
                hello=getchar();/*Take up the return*/
                comp=fopen(file,"rb");/*Open compare file and rb mean read in binary*/
                printf("\n\nWORKING!!!!\n\n");/*Know if it is working*/

                while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
                {       nf = nf + 1;/*Frame counter*/
                        while (fread(in1.data,(nwpf[fi]*nbpw[fi]/8),1,comp))/*Read in
 data from compared file and store it in in1.data*/
                        {
                                for(j=0; j<nwpf[fi]*nbpw[fi]/8; j++)/*Looping through the
 frame*/
                                {
                                        if(in1.data[j]!=in.data[j])/*Checking if the two files
 match 8 bits at a time*/
                                        {
                                                printf("\nERROR in frame %ld and in
 %ld",nf,j);/*Print error if the data doesn't match*/

        printf("\n%02x\t%02x\n",in.data[j],in1.data[j]);/*Print in hex*/
```

```
                                                printf("\nDo you want to return to menu
(Y,N)\n");/*If the is a error this will let you return to the begining*/
                                                hello=getchar();/*Take up the return*/
                                                if(hello=='y'||hello=='Y')
                                                {
                                                        goto again;/*If you want to know
where this goes to look for again at the bottom*/
                                                }
                                        }
                                }break;/*This breaks the while loop for the compare file so
that it read in a new frame*/
                        }
                }
        printf("\nDONE COMPARING %s AND %s\n\n",filename[fi],file);/*This prints the file
names that were compared*/
        printf ("Successful EndOfFile reached %ld Minor Frames.\n\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);/*Close the data file*/
        fclose(comp);/*Close the compare file*/
        nf = 0;
        again:/*This is where that goto function goes to*/
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*This prints menu of the
files*/
        printf ("  Enter the file Number\n");
        printf ("  or <ctrl> c to end: \n");
        }
} /* End Main */
```

COMP3CD

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data, word, interval, rs, sfid;
unsigned  int z=0,i,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int  nfrms[numfiles], wow[lengfiln];
unsigned long int j=0,nf=0, fsp[numfiles], nbps[numfiles];
char response, excel[32],newfile[lengfiln],file[lengfiln];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
                unsigned int b15: 2;
```

```
            unsigned int b16: 2;
        } ;
        struct lnibles{
            unsigned int n1 : 4;
            unsigned int n2 : 4;
            unsigned int n3 : 4;
            unsigned int n4 : 4;
            unsigned int n5 : 4;
            unsigned int n6 : 4;
            unsigned int n7 : 4;
            unsigned int n8 : 4;
        };
        struct lbytes{
            unsigned int n1 : 8;
            unsigned int n2 : 8;
            unsigned int n3 : 8;
            unsigned int n4 : 8;
        };
        union
        {
            struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[2048];
            struct bytes    byte[2048];
            unsigned  char  data[2048];
        } in ;
            union
        {
            struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[2048];
            struct bytes    byte[2048];
            unsigned  char  data[2048];
        } in1 ;
        union
        {
            struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
            struct lnibles   nible[4096];
            struct lbytes    byte[4096];
            unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp, *comp;/*File pointers*/

void main()
{
        i = 0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",
```

```
        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);/*Closed cinput.dat*/

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                        newfile[0]='e';
                        newfile[1]=':';
                        z=0;
                         do
                         {
                                newfile[2+z]=filename[fi][z];/*This puts 'e:' in front of the file
 name so it can read it off the cd*/
                                z++;
                         }while(z<=10);z=0;

                f=fopen(newfile,"rb");/* Open selected data file */
                printf("\nType in the path and file name you would like to compare that to:\n");
                scanf("%s",file);
                hello=getchar();/*Take up the return*/
                comp=fopen(file,"rb");/*Open compare file and rb mean read in binary*/
                printf("\n\nWORKING!!!!\n\n");/*Know if it is working*/
```

```
            while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
            {       nf = nf + 1;/*Frame counter*/
                        while (fread(in1.data,(nwpf[fi]*nbpw[fi]/8),1,comp))/*Read in
data from compared file and store it in in1.data*/
                        {
                                for(j=0; j<nwpf[fi]*nbpw[fi]/8; j++)/*Looping through the
frame*/
                                {
                                        if(in1.data[j]!=in.data[j])/*Checking if the two files
match 8 bits at a time*/
                                        {
                                                printf("\nERROR in frame %ld and in
%ld",nf,j);/*Print error if the data doesn't match*/

      printf("\n%02x\t%02x\n",in.data[j],in1.data[j]);/*Print in hex*/

                                                printf("\nDo you want to return to menu
(Y,N)\n");/*If the is a error this will let you return to the begining*/
                                                hello=getchar();/*Take up the return*/
                                                if(hello=='y'||hello=='Y')
                                                {
                                                        goto again;/*If you want to know
where this goes to look for again at the bottom*/
                                                }
                                        }
                                }break;/*This breaks the while loop for the compare file so
that it read in a new frame*/
                        }
                }
        printf("\nDONE COMPARING %s AND %s\n\n",filename[fi],file);/*This prints the file
names that were compared*/
        printf ("Successful EndOfFile reached %ld Minor Frames.\n\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);/*Close the data file*/
        fclose(comp);/*Close the compare file*/
        nf = 0;
        again:/*This is where that goto function goes to*/
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*This prints menu of the
files*/
        printf ("  Enter the file Number\n");
        printf ("  or <ctrl> c to end: \n");
        }
} /* End Main */
```

COUNT12

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int b=0,i, j, z=0,x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles],
sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int nf=0,fsp[numfiles],
nbps[numfiles],afb1[256],counter1[256],afb2[256],counter2[256],afb3[256],counter3[256];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30], newfile[40];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
```

```
                unsigned int b14: 2;
                unsigned int b15: 2;
                unsigned int b16: 2;
        } ;
        struct lnibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
                unsigned int n3 : 4;
                unsigned int n4 : 4;
                unsigned int n5 : 4;
                unsigned int n6 : 4;
                unsigned int n7 : 4;
                unsigned int n8 : 4;
        };
        struct lbytes{
                unsigned int n1 : 8;
                unsigned int n2 : 8;
                unsigned int n3 : 8;
                unsigned int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{ i = 0;
        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
```

```
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                printf("What is the file you want counted:\n");
                scanf("%s",newfile);
                printf("\nWORKING!!!!\n");

                f=fopen(newfile,"rb");/* Open selected data file */
                        n=0;
                        z=0;
                        do
                        {
                                afb2[z]=z;/*This puts numbers into the array for comparing*/
                                counter2[z]=0;/*This puts zeros in the array to set the count to
 zero*/
                                z++;
                        }while(z<=15);z=0;/*Puts numbers up to 15 in the array*/
                        do
                        {
                                afb1[n]=n;/*This puts numbers into the array for comparing*/
                                counter1[n]=0;/*This puts zeros in the array to set the count to
 zero*/
                                n++;
                        }while(n<=255);n=0;/*Puts numbers up to 255 in the array*/
                while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
                {       nf = nf + 1;/*Frame counter*/
                        j=0;
```

```
                    for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                    {
                        do
                        {
                            if(in.data[j]==afb1[n])/*Check if the numbers
match*/
                            {
                                counter1[n]+=1;/*counter if they
match*/
                                break;/*Breaks once counted*/
                            }
                            n++;
                        }while(n<=255);n=0;
                    }
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data into nibles*/
                        {   out.nible[i]  .n3 = in.nible[j]  .n2;
                            out.nible[i]  .n2 = in.nible[j]  .n1;
                            out.nible[i]  .n1 = in.nible[j+1].n2;
                            out.nible[i+1].n3 = in.nible[j+1].n1;
                            out.nible[i+1].n2 = in.nible[j+2].n2;
                            out.nible[i+1].n1 = in.nible[j+2].n1;
                            i=i+2;
                        }
                        for (j=0; j<nwpf[fi]*nbpw[fi]/12; j++)/*Loop through the
frame of 12 bit words*/
                        { z=0;
                            do
                            {
                                if(out.nible[j].n1==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                {
                                    counter2[z]+=1;
                                }
                                if(out.nible[j].n2==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                {
                                    counter2[z]+=1;
                                }
                                if(out.nible[j].n3==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                {
                                    counter2[z]+=1;
                                }
                            z++;
                            }while(z<=15);z=0;
                        }
```

```
                }
        printf("\n%s DONE!!!!\n",newfile);
        printf("\nOutput file name for the two hex #: ");
        scanf("%s",newfile1);
        printf("\nOutput file name for the one hex #: ");
        scanf("%s",newfile2);
        lp2=fopen(newfile2,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp2,"%s\n",filename[fi]);/*Prints file name to the output file*/
        lp1=fopen(newfile1,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp1,"%s\n",filename[fi]);/*Prints file name to the output file*/
        n=0;
        do
        {
                fprintf(lp1,"%lx\tCount\t%ld\n",afb1[n],(long)counter1[n]);/*Printing to the
output file*/
                n++;
        }while(n<=255);n=0;
        z=0;
        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long)counter2[z]);/*Printing to the
output file*/
                z++;
        }while(z<=15);z=0;
        printf ("\nSuccessful EndOfFile reached %ld Minor Frames.\n\n",(long) nf);/* Cleanup,
close-up, and start over */

        fclose(f);
        fclose(lp1);
        fclose(lp2);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Prints menu of files*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

FRAMAV

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int z=0, i, j,c=0, nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int wow[lengfiln];
unsigned long int nfrms[numfiles],nf=0, k=0, fsp[numfiles], nbps[numfiles], abf[lengfiln],
abf1[lengfiln], abf2[lengfiln];
char response, excel[32], outfile[40],newfile[40];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
```

```
        unsigned int b15: 2;
        unsigned int b16: 2;
    } ;
    struct lnibles{
        unsigned int n1 : 4;
        unsigned int n2 : 4;
        unsigned int n3 : 4;
        unsigned int n4 : 4;
        unsigned int n5 : 4;
        unsigned int n6 : 4;
        unsigned int n7 : 4;
        unsigned int n8 : 4;
    };
    struct lbytes{
        unsigned int n1 : 8;
        unsigned int n2 : 8;
        unsigned int n3 : 8;
        unsigned int n4 : 8;
    };
    union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in ;
    union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in1 ;
    union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
        unsigned  char  data[2048];
    } in2 ;
    union
    {
        struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes   byte[2048];
```

```
                unsigned char data[2048];
        } in3 ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp, *fout;

void main()
{

        for (i = 0; i < pltln; i++) plot[i] = ' ';/* Initialize array */

        i = 0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames =%ld
\n",nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
```

```
            nb=0;i=1;
            while ((sfl[fi]-sff[fi])>i)        {nb=nb+1;i=2*i;}
            rs = nbpw[fi]-sfido[fi]-nb+1;
                    newfile[0]='e';
                    newfile[1]=':';
                    z=0;
                     do
                     {
                             newfile[2+z]=filename[fi][z];
                             z++;
                     }while(z<=10);z=0;
            printf("\nWORKING!!!!\n");
            f=fopen(newfile,"rb");/* Open selected data file */
             for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                    {
                                    abf[j]=0;
                    }
            while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
            {       nf = nf + 1;
                      for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                      {
                              abf[j]=abf[j]+in.data[j];
                      }
            }
            printf("Type in path and file name for the output:\n");
            scanf("%s",outfile);
            fout=fopen(outfile,"w");
        for (k=0; k<nwpf[fi]*nbpw[fi]/8; k++)
        {
            abf1[k]=abf[k]/nf;
            fprintf(fout,"%ld",abf1[k]);
        }
        printf ("Successful EndOfFile reached %d Minor Frames.\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);
        fclose(fout);
        if (result > 2)fclose(fo);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

FRAMINU1

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, cool, abf1[lengfiln], abf4[lengfiln], abf2[lengfiln], bye, a, u;
unsigned int data,word, interval, rs, sfid, joke;
unsigned  int good, k=0,i=0, z=0, j, nwpf[numfiles], nbpw[numfiles], sfidw[numfiles],
sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int b=0, wow[lengfiln], abf3[lengfiln];
unsigned long int nfrms[numfiles], nf=0, abf5[500], fsp[numfiles], nbps[numfiles];
long int boo=0, one, negative[300], negcount[300], h, m, p, positive[300], poscount[300];
char response, check, okay, excel[32], newfile[40], file[40], addfile[40], could[40];
char hello, song;
long int l=0, count=0, c=0, v, saw, hi, fi, try, abf[lengfiln];
struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
```

```
            unsigned int b12: 2;
            unsigned int b13: 2;
            unsigned int b14: 2;
            unsigned int b15: 2;
            unsigned int b16: 2;
        } ;
        struct lnibles{
            unsigned int n1 : 4;
            unsigned int n2 : 4;
            unsigned int n3 : 4;
            unsigned int n4 : 4;
            unsigned int n5 : 4;
            unsigned int n6 : 4;
            unsigned int n7 : 4;
            unsigned int n8 : 4;
        };
        struct lbytes{
            unsigned int n1 : 8;
            unsigned int n2 : 8;
            unsigned int n3 : 8;
            unsigned int n4 : 8;
        };
        union
        {
            struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[6048];
            struct bytes    byte[6048];
            unsigned  char  data[6048];
        } in ;
        union
        {
            struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[6048];
            struct bytes    byte[6048];
            signed  char  data[6048];
        } in4;
        union
        {
            struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[6048];
            struct bytes    byte[6048];
            unsigned  char  data[6048];
        } in1 ;
        union
        {
```

```
                struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[6048];
                struct bytes    byte[6048];
                unsigned  char  data[6048];
        } in2 ;
        union
        {
                struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[6048];
                struct bytes    byte[6048];
                signed  char  data[6048];
        } in3 ;
        union
        {
                struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[6048];
                struct bytes.   byte[6048];
                signed  char  data[6048];
        } in5 ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *g, *fin, *fo, *lp, *fout, *added, *frame;

void main()
{
                printf("\nMENU:");
                printf("\nADD->'a'");
                printf("\nMINUS->'m'");
                printf("\nAVG->'v'\n:");
                response=getchar();
                hello=getchar();
                check=response;
                okay=response;
        i = 0;
        c=0;
        count=0;
        l=0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",
```

```
        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                l=0;
                if(response=='m'||response=='M')
                {
                newfile[0]='e';
                newfile[1]=':';
                        b=0;
                         do
                         {
                                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
 name so it can read it off the cd*/
                                b++;
                         }while(b<=10);b=0;
                         f=fopen(newfile,"rb");/* Open selected data file */
                         fo=fopen("frame.dat","wb");
                         fout=fopen("new.dat","wb");
                         fread(in1.data,(nwpf[fi]*nbpw[fi]),1,f);
                         fwrite(in1.data,(nwpf[fi]*nbpw[fi]),1,fo);
```

```
fclose(fo);
v=0;
l=0;
c=0;
count=1;
while(fread(&in2.data[0],sizeof(int),1,f))
{
                        in4.data[0]=(in1.data[c]-in2.data[0])*(-1);
                        fwrite(&in4.data[0],sizeof(in4.data[0]),1,fout);
                        in.data[0]=in1.data[c]+in4.data[0];
                        if(in4.data[0]<0)
                        {
                                do
                                {
                                        if(negative[m]==in4.data[0])
                                        {
                                                negcount[m]+=1;
                                                h++;
                                        }
                                        l++;
                                } while(l<=m);l=0;
                                if(h!=1)
                                        {
                                                negative[m+1]=in4.data[0];
                                                negcount[m]+=1;
                                        }
                                h=0;
                        }
                        if(in4.data[0]>=0)
                        {
                                do
                                {
                                        if(positive[p]==in4.data[0])
                                        {
                                                poscount[p]+=1;
                                                h++;
                                        }
                                        l++;
                                } while(l<=p);l=0;
                                if(h!=1)
                                        {
                                                positive[p+1]=in4.data[0];
                                                poscount[p]+=1;
                                        }
                                h=0;
                        }
                        if(in.data[0]!=in2.data[0])
                        {
```

```
                                        printf("(Error) in %d = in1 %d + in4
%d\n",in.data[0],in1.data[c],in4.data[0]);
                                        hello=getchar();
                                }
                                if(count==(nwpf[fi]*nbpw[fi]/8))
                                {
                                        count=1;
                                        l++;
                                }
                                c++;
                                count++;
                                if(c==(nwpf[fi]*nbpw[fi]))
                                {
                                        c=0;
                                        printf(".");
                                }
                                in5.data[0]=(in1.data[c]-in2.data[1])*(-1);
                                fwrite(&in5.data[0],sizeof(in5.data[0]),1,fout);
                                in.data[0]=in1.data[c]+in5.data[0];
                                if(in.data[0]!=in2.data[1])
                                {
                                        printf("(Error) in %d = in1 %d + in4
%d\n",in.data[0],in1.data[c],in4.data[0]);
                                        hello=getchar();
                                }
                                if(count==(nwpf[fi]*nbpw[fi]/8))
                                {
                                        count=1;
                                        l++;
                                }
                                c++;
                                count++;
                                if(c==(nwpf[fi]*nbpw[fi]))
                                {
                                        c=0;
                                        printf(".");
                                }
                        hello=getchar();
                }
                printf("\nThe minus output files are frame.dat and new.dat in this
directory\n\n");
                printf ("Successful EndOfFile reached %ld Minor Frames.\n",(long)l);
                fclose(f);
                fclose(fout);
                count=0;
        }
        l=0;
        c=0;
        boo=0;
```

```
if(response=='a'||response=='A')
{
        fclose(f);
        newfile[0]='e';
        newfile[1]=':';
        b=0;
        do
        {
                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off the cd*/
                b++;
        }while(b<=10);b=0;
        f=fopen(newfile,"rb");
        printf("\nWhat is the file you what to add to:");
        scanf("%s",could);
        printf("\nOutput file name for the added data: ");
        scanf("%s",addfile);
        hello=getchar();
        g=fopen(could,"rb");
        added=fopen(addfile,"wb");
        frame=fopen("frame.dat","rb");
        fread(in.data,(nwpf[fi]*nbpw[fi]),1,frame);
        printf("\nWas the frames minused: (y,n) ");
        song=getchar();
        hello=getchar();
        if(song=='y')
        {
                fwrite(in.data,(nwpf[fi]*nbpw[fi]),1,added);
                fread(in5.data,(nwpf[fi]*nbpw[fi]),1,f);
        }
        fclose(frame);
        l=0;
        c=1;
        while(fread(&in3.data[0],sizeof(in3.data[0]),1,g))
        {
                in2.data[0]=in3.data[0]+in.data[l];
                fread(&in1.data[0],sizeof(in1.data[0]),1,f);
                if(in2.data[0]!=in1.data[0])
                {
                        printf("frame %ld\n",c);
                        printf("word %d\n",l);
                        printf("\n(Error) in1 %d  in2
%d\n",in1.data[0],in2.data[0]);
                        printf("\nIN3 %d + in %d = in2
%d\n",in3.data[0],in.data[l],in2.data[0]);
                        hello=getchar();
                }
                l++;
                if(l==(nwpf[fi]*nbpw[fi]))
```

```
                {
                        printf(".");
                        l=0;
                        c++;
                }
                fwrite(&in2.data[0],sizeof(in2.byte[0]),1,added);
        }
printf("\nThe added output file %s is done!\n",addfile);
printf("\nNumber of frames %ld\n",(long)c);
fclose(added);
fclose(g);
fclose(f);
fclose(frame);
}
l=0;
c=0;
if(response=='v'||response=='v')
{
newfile[0]='e';
newfile[1]=':';
file[0]='e';
file[1]=':';
        b=0;
        do
        {
                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off the cd*/
                file[2+b]=filename[fi][b];
                b++;
        }while(b<=10);b=0;
        f=fopen(newfile,"rb");/* Open selected data file */
        fo=fopen("frame.dat","wb");
        fout=fopen("new.dat","wb");
        k=0;
        j=0;
        while(fread(in1.data,sizeof(in.byte[0]),1,f))
        {
                        if(j<=1)
                        {
                                abf[k]=in1.data[0];
                        }
                        if(j>1)
                        {
                                abf[k]=in1.data[0]+abf[k];
                        }
                        k++;
                        if(k==(nwpf[fi]*nbpw[fi]))
                        {
                                printf(".");
```

```
                                        k=0;
                                        j++;
                                }
                }
                        do
                        {
                                in2.data[v]=abf[v]/j;
                                v++;
                        }while(v<=(nwpf[fi]*nbpw[fi]));v=0;
                        fwrite(in2.data,(nwpf[fi]*nbpw[fi]),1,fo);
                        c=0;
                        k=0;
                        fclose(fo);
                        fclose(f);
                        g=fopen(file,"rb");
                while(fread(&in.data[0],sizeof(int),1,g))
                {
                                in4.data[0]=(in2.data[k]-in.data[0])*(-1);
                                fwrite(&in4.data[0],sizeof(in4.data[0]),1,fout);
                                k++;
                                in5.data[0]=(in2.data[k]-in.data[1])*(-1);
                                fwrite(&in5.data[0],sizeof(in5.data[0]),1,fout);
                                k++;
                                if(k==(nwpf[fi]*nbpw[fi]/8))
                                {
                                        l++;
                                }
                                if(k==(nwpf[fi]*nbpw[fi]))
                                { printf(".");
                                        k=0;
                                }
                        }
                        fclose(g);
                        fclose(fout);
                        fclose(frame);
                        fclose(f);
                        fclose(fo);
                        printf("\nThe average frame in is frame.dat and the minus data is in
new.dat \n\n");
                        printf ("\nSuccessful EndOfFile reached %ld Minor
Frames.\n",(long)l);/* Cleanup, close-up, and start over */

                }
                else
                {}
        k=0;
        boo=0;
        j=0;
        c=0;
```

```
        nf = 0;
        l=0;
        printf("\nMENU:");
        printf("\nADD->'a'");
        printf("\nMINUS->'m'");
        printf("\nAVG->'v'\n:");
        scanf("%c",&response);
        hello=getchar();
        count=0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); } /*Print out menu*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

FRAMMINU

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, cool, abf1[lengfiln], abf4[lengfiln], abf2[lengfiln], bye, a, u;
unsigned int data,word, interval, rs, sfid, joke;
unsigned  int good, k=0,i=0, z=0, j, nwpf[numfiles], nbpw[numfiles], sfidw[numfiles],
sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int b=0, wow[lengfiln], abf3[lengfiln];
unsigned long int nfrms[numfiles], nf=0, abf5[1000], fsp[numfiles], nbps[numfiles];
long int boo=0, one;
char response, check, okay, excel[32], newfile[40], file[40], addfile[40], could[40];
char hello, song;
long int l=0, count=0, c=0, v, saw, hi, fi, try, abf[lengfiln];
struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
```

```
            unsigned int b12: 2;
            unsigned int b13: 2;
            unsigned int b14: 2;
            unsigned int b15: 2;
            unsigned int b16: 2;
        } ;
        struct lnibles{
            unsigned int n1 : 4;
            unsigned int n2 : 4;
            unsigned int n3 : 4;
            unsigned int n4 : 4;
            unsigned int n5 : 4;
            unsigned int n6 : 4;
            unsigned int n7 : 4;
            unsigned int n8 : 4;
        };
        struct lbytes{
            unsigned int n1 : 8;
            unsigned int n2 : 8;
            unsigned int n3 : 8;
            unsigned int n4 : 8;
        };
        union
        {
            struct tbits   tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[6048];
            struct bytes   byte[6048];
            unsigned  char  data[6048];
        } in ;
        union
        {
            struct tbits   tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[6048];
            struct bytes   byte[6048];
            signed  char  data[6048];
        } in4;
        union
        {
            struct tbits   tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[6048];
            struct bytes   byte[6048];
            unsigned  char  data[6048];
        } in1 ;
        union
        {
```

```
                struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[6048];
                struct bytes    byte[6048];
                unsigned  char  data[6048];
        } in2 ;
        union
        {
                struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[6048];
                struct bytes    byte[6048];
                signed  char  data[6048];
        } in3 ;
        union
        {
                struct tbits    tbit[6048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[6048];
                struct bytes . byte[6048];
                signed  char  data[6048];
        } in5 ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *g, *fin, *fo, *lp, *fout, *added, *frame;

void main()
{
                printf("\nMENU:");
                printf("\nADD->'a'");
                printf("\nMINUS->'m'");
                printf("\nAVG->'v'\n:");
                response=getchar();
                hello=getchar();
                check=response;
                okay=response;
        i = 0;
        c=0;
        count=0;
        l=0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",
```

```
        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                l=0;
                if(response=='m'||response=='M')
                {
                newfile[0]='e';
                newfile[1]=':';
                        b=0;
                         do
                         {
                                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off the cd*/
                                b++;
                         }while(b<=10);b=0;
                         f=fopen(newfile,"rb");/* Open selected data file */
                         fo=fopen("frame.dat","wb");
                         fout=fopen("new.dat","wb");
                         fread(in1.data,(nwpf[fi]*nbpw[fi]),1,f);
                         fwrite(in1.data,(nwpf[fi]*nbpw[fi]),1,fo);
```

```
                fclose(fo);
                v=0;
                l=0;
                c=0;
                count=1;
                while(fread(&in2.data[0],sizeof(int),1,f))
                {
                                in4.data[0]=(in1.data[c]-in2.data[0])*(-1);
                                fwrite(&in4.data[0],sizeof(in4.data[0]),1,fout);
                                in.data[0]=in1.data[c]+in4.data[0];
                                if(in.data[0]!=in2.data[0])
                                {
                                        printf("(Error) in %d = in1 %d + in4
%d\n",in.data[0],in1.data[c],in4.data[0]);
                                        hello=getchar();
                                }
                                if(count==(nwpf[fi]*nbpw[fi]/8))
                                {
                                        count=1;
                                        l++;
                                }
                                c++;
                                count++;
                                if(c==(nwpf[fi]*nbpw[fi]))
                                {
                                        c=0;
                                        printf(".");
                                }
                                in5.data[0]=(in1.data[c]-in2.data[1])*(-1);
                                fwrite(&in5.data[0],sizeof(in5.data[0]),1,fout);
                                in.data[0]=in1.data[c]+in5.data[0];
                                if(in.data[0]!=in2.data[1])
                                {
                                        printf("(Error) in %d = in1 %d + in4
%d\n",in.data[0],in1.data[c],in4.data[0]);
                                        hello=getchar();
                                }
                                if(count==(nwpf[fi]*nbpw[fi]/8))
                                {
                                        count=1;
                                        l++;
                                }
                                c++;
                                count++;
                                if(c==(nwpf[fi]*nbpw[fi]))
                                {
                                        c=0;
                                        printf(".");
                                }
```

```
                }
                printf("\nThe minus output files are frame.dat and new.dat in this
directory\n\n");
                printf ("Successful EndOfFile reached %ld Minor Frames.\n",(long)l);
                fclose(f);
                fclose(fout);
                count=0;
        }
        l=0;
        c=0;
        boo=0;
        if(response=='a'||response=='A')
        {
                fclose(f);
                newfile[0]='e';
                newfile[1]=':';
                b=0;
                 do
                 {
                        newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off the cd*/
                        b++;
                 }while(b<=10);b=0;
                f=fopen(newfile,"rb");
                printf("\nWhat is the file you what to add to:");
                scanf("%s",could);
                printf("\nOutput file name for the added data: ");
                scanf("%s",addfile);
                hello=getchar();
                g=fopen(could,"rb");
                added=fopen(addfile,"wb");
                frame=fopen("frame.dat","rb");
                fread(in.data,(nwpf[fi]*nbpw[fi]),1,frame);
                printf("\nWas the frames minused: (y,n) ");
                song=getchar();
                hello=getchar();
                if(song=='y')
                {
                        fwrite(in.data,(nwpf[fi]*nbpw[fi]),1,added);
                        fread(in5.data,(nwpf[fi]*nbpw[fi]),1,f);
                }
                fclose(frame);
                l=0;
                c=1;
                while(fread(&in3.data[0],sizeof(in3.data[0]),1,g))
                {
                        in2.data[0]=in3.data[0]+in.data[l];
                        fread(&in1.data[0],sizeof(in1.data[0]),1,f);
                        if(in2.data[0]!=in1.data[0])
```

```
                        {
                                printf("frame %ld\n",c);
                                printf("word %d\n",l);
                                printf("\n(Error) in1 %d  in2
%d\n",in1.data[0],in2.data[0]);
                                printf("\nIN3 %d + in %d = in2
%d\n",in3.data[0],in.data[l],in2.data[0]);
                                hello=getchar();
                        }
                        l++;
                        if(l==(nwpf[fi]*nbpw[fi]))
                        {
                                printf(".");
                                l=0;
                                c++;
                        }
                        fwrite(&in2.data[0],sizeof(in2.byte[0]),1,added);
                }
        printf("\nThe added output file %s is done!\n",addfile);
        printf("\nNumber of frames %ld\n",(long)c);
        fclose(added);
        fclose(g);
        fclose(f);
        fclose(frame);
        }
        l=0;
        c=0;
        if(response=='v'||response=='v')
        {
        newfile[0]='e';
        newfile[1]=':';
        file[0]='e';
        file[1]=':';
                b=0;
                 do
                 {
                        newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off the cd*/
                        file[2+b]=filename[fi][b];
                        b++;
                 }while(b<=10);b=0;
                 f=fopen(newfile,"rb");/* Open selected data file */
                 fo=fopen("frame.dat","wb");
                 fout=fopen("new.dat","wb");
                 k=0;
                 j=0;
                 while(fread(in1.data,sizeof(in.byte[0]),1,f))
                 {
                                        if(j<=1)
```

```
                {
                        abf[k]=in1.data[0];
                }
                if(j>1)
                {
                        abf[k]=in1.data[0]+abf[k];
                }
                k++;
                if(k==(nwpf[fi]*nbpw[fi]))
                {
                        printf(".");
                        k=0;
                        j++;
                }
}
        do
        {
                in2.data[v]=abf[v]/j;
                v++;
        }while(v<=(nwpf[fi]*nbpw[fi]));v=0;
        fwrite(in2.data,(nwpf[fi]*nbpw[fi]),1,fo);
        c=0;
        k=0;
        fclose(fo);
        fclose(f);
        g=fopen(file,"rb");
while(fread(&in.data[0],sizeof(int),1,g))
{
                in4.data[0]=(in2.data[k]-in.data[0])*(-1);
                fwrite(&in4.data[0],sizeof(in4.data[0]),1,fout);
                k++;
                in5.data[0]=(in2.data[k]-in.data[1])*(-1);
                fwrite(&in5.data[0],sizeof(in5.data[0]),1,fout);
                k++;
                if(k==(nwpf[fi]*nbpw[fi]/8))
                {
                        l++;
                }
                if(k==(nwpf[fi]*nbpw[fi]))
                { printf(".");
                        k=0;
                }
        }
        fclose(g);
        fclose(fout);
        fclose(frame);
        fclose(f);
        fclose(fo);
```

```
                        printf("\nThe average frame in is frame.dat and the minus data is in
new.dat \n\n");
                        printf ("\nSuccessful EndOfFile reached %ld Minor
Frames.\n",(long)l);/* Cleanup, close-up, and start over */

                    }
                    else
                    {}
        k=0;
        boo=0;
        j=0;
        c=0;
        nf = 0;
        l=0;
        printf("\nMENU:");
        printf("\nADD->'a'");
        printf("\nMINUS->'m'");
        printf("\nAVG->'v'\n:");
        scanf("%c",&response);
        hello=getchar();
        count=0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Print out menu*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

IMPROVE

```
/*
*-------------------------------------------------------------------
*
*       Name:                                   Display_Data.c
*       Creation Date:                  09-Sep-97
*       Description:                    Display Data from Telemetry Sample Data Sets
*       Author:                                 Vernon Diekmann, TYBRIN
*       Target Machine/OS/Compiler:     Intel Pentium/Windows 95/NT/MS Visual C++ 5.0
*-------------------------------------------------------------------
*
*       Revision History
*
*       Initials Date           Description
*       -------- ----           -----------
*       VD              09/09/97        Initial version
*       VD              10/10/97        Added 12 and 16 bit
*       VD              01/15/98        Added 24 bit and read byte records
*       VD              01/28/98        Input the file name for the data.dat file.
*
-------------------------------------------------------------------
*/
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int i, j, nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int nfrms[numfiles], wow[lengfiln];
unsigned long int nf=0, fsp[numfiles], nbps[numfiles];
char response, excel[32];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
```

```
};
struct bytes{
        unsigned int n1 : 8;
};
struct ltbits
{
        unsigned int b1 : 2;
        unsigned int b2 : 2;
        unsigned int b3 : 2;
        unsigned int b4 : 2;
        unsigned int b5 : 2;
        unsigned int b6 : 2;
        unsigned int b7 : 2;
        unsigned int b8 : 2;
        unsigned int b9 : 2;
        unsigned int b10: 2;
        unsigned int b11: 2;
        unsigned int b12: 2;
        unsigned int b13: 2;
        unsigned int b14: 2;
        unsigned int b15: 2;
        unsigned int b16: 2;
} ;
struct lnibles{
        unsigned int n1 : 4;
        unsigned int n2 : 4;
        unsigned int n3 : 4;
        unsigned int n4 : 4;
        unsigned int n5 : 4;
        unsigned int n6 : 4;
        unsigned int n7 : 4;
        unsigned int n8 : 4;
};
struct lbytes{
        unsigned int n1 : 8;
        unsigned int n2 : 8;
        unsigned int n3 : 8;
        unsigned int n4 : 8;
};
union
{
        struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes    byte[2048];
        unsigned  char  data[2048];
} in ;
union
{
```

```
            struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
            struct lnibles   nible[4096];
            struct lbytes    byte[4096];
            unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp;

void main()
{
        for (i = 0; i < pltln; i++) plot[i] = ' ';/* Initialize array */

        i = 0;

        fin=fopen("cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
 once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
 %d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
 First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                printf ("  Enter\nF to print each frame or \nW to plot a single word: \n");
```

```
response = getchar();

if (response == 'w' || response == 'W')
{
        printf ("  Enter the\n Word, Subframe, Interval to display: \n");
        result = scanf("%d%d%d",&word,&subframe,&interval);
        hello = getchar();

        printf ("  Enter 'c' to continue or type the path and file name for the data of
the word selected. \n");
        result = scanf("%s",&excel);
}
f=fopen(&filename[fi][0],"rb");/* Open selected data file */

result = strlen(&excel[0]);

if(result > 2)
        fo=fopen(&excel[0],"w");/* Open option data output file for excel */

while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
{       nf = nf + 1;
        if (response == 'f' || response == 'F')/* Display each minor frame */
        {       printf ("\nFrame=%d\n",nf);
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                {
                        printf("%02x",in.data[j]);
                }
                printf("\n");
                printf ("Enter to continue <ctrl>C to quit");
                hello=getchar();
        }
        else if (response == 'w' || response == 'W') /* Display selected data
parameter */
        {
                if (nbpw[fi]==8)
                {
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=1)/* Unravel packed 8
bit data */
                        {
                                out.byte[i].n1  = in.byte[j].n1;
                        }
                }
                if (nbpw[fi]==10)
                {
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=5)/* Unravel packed 10
bit data */
                        {
```

```
                        out.tbit[i].b5   = in.tbit[j].b4;
                        out.tbit[i].b4   = in.tbit[j].b3;
                        out.tbit[i].b3   = in.tbit[j].b2;
                        out.tbit[i].b2   = in.tbit[j].b1;
                        out.tbit[i].b1   = in.tbit[j+1].b4;
                        out.tbit[i+1].b5 = in.tbit[j+1].b3;
                        out.tbit[i+1].b4 = in.tbit[j+1].b2;
                        out.tbit[i+1].b3 = in.tbit[j+1].b1;
                        out.tbit[i+1].b2 = in.tbit[j+2].b4;
                        out.tbit[i+1].b1 = in.tbit[j+2].b3;
                        out.tbit[i+2].b5 = in.tbit[j+2].b2;
                        out.tbit[i+2].b4 = in.tbit[j+2].b1;
                        out.tbit[i+2].b3 = in.tbit[j+3].b4;
                        out.tbit[i+2].b2 = in.tbit[j+3].b3;
                        out.tbit[i+2].b1 = in.tbit[j+3].b2;
                        out.tbit[i+3].b5 = in.tbit[j+3].b1;
                        out.tbit[i+3].b4 = in.tbit[j+4].b4;
                        out.tbit[i+3].b3 = in.tbit[j+4].b3;
                        out.tbit[i+3].b2 = in.tbit[j+4].b2;
                        out.tbit[i+3].b1 = in.tbit[j+4].b1;
                        i=i+4;
                }
        }

        if (nbpw[fi]==12)
        {
                i=0;
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data */
                {       out.nible[i]  .n3 = in.nible[j]  .n2;
                        out.nible[i]  .n2 = in.nible[j]  .n1;
                        out.nible[i]  .n1 = in.nible[j+1].n2;
                        out.nible[i+1].n3 = in.nible[j+1].n1;
                        out.nible[i+1].n2 = in.nible[j+2].n2;
                        out.nible[i+1].n1 = in.nible[j+2].n1;
                        i=i+2;
                }
        }
        if (nbpw[fi]==16)
        {
                i=0;
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=2)/* Unravel packed 16
bit data */
                {       out.byte[i].n2 = in.byte[j]  .n1;
                        out.byte[i].n1 = in.byte[j+1].n1;
                        i=i+1;
                }
        }
        if (nbpw[fi]==24)
```

```
                {
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 24
bit data */
                        {       out.byte[i].n3 = in.byte[j]  .n1;
                                out.byte[i].n2 = in.byte[j+1].n1;
                                out.byte[i].n1 = in.byte[j+2].n1;
                                i=i+1;
                        }
                }
                if ((sfidw[fi] >= 0) & ( interval != 0) & (subframe >= 0))/* Handle
sub frame data */
                {       sfid = out.data[sfidw[fi]-1];
                        sfid = sfid >> rs;
                        for (j=0; j<(sfl[fi]-sff[fi]+1); j+=interval)
                        {       if ((subframe+j) == sfid)
                                {       data = (unsigned)out.data[word-1]/pltln;
                                        data = (unsigned)out.data[word-1]-
data*pltln;
                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2)fprintf (fo,"%d\t%d\n",nf,
out.data[word-1]);
                                }
                        }
                }
                else if ((subframe == -1) & ( interval != 0))/* Handle super
commutated data */
                {       for (j=0; j<=nwpf[fi]; j+=interval)

                                {       data = (unsigned)out.data[word-1+j]/pltln;
                                        data = (unsigned)out.data[word-1+j]-
data*pltln;
                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2 )fprintf(fo,"%d\t%d\n",nf,
out.data[word-1+j]);
                                }

                }
                else /* Just plain old data */
                {       data = (unsigned)out.data[word-1]/pltln;
                        data = (unsigned)out.data[word-1]-data*pltln;
```

```
                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2)fprintf (fo,"%d\t%d\n",nf, out.data[word-1]);
                                }
                        }
                        else
                        {       /* Do nothing */
                        }
                }
        printf ("Successful EndOfFile reached %d Minor Frames.\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);

        if (result > 2)fclose(fo);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

NEGATIVE

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int b=0,i, j, z=0,x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles],
sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int nf=0,fsp[numfiles],
nbps[numfiles],afb1[256],counter1[256],afb2[256],counter2[256],afb3[256],counter3[256];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30], newfile[40];
long negative1[2000],negative2[2000],negative3[2000], negative4[2000];
        struct tbits
        {
                signed int b1 : 2;
                signed int b2 : 2;
                signed int b3 : 2;
                signed int b4 : 2;
        } ;
        struct nibles{
                signed int n1 : 4;
                signed int n2 : 4;
        };
        struct bytes{
                signed int n1 : 8;
        };
        struct ltbits
        {
                signed int b1 : 2;
                signed int b2 : 2;
                signed int b3 : 2;
                signed int b4 : 2;
                signed int b5 : 2;
                signed int b6 : 2;
                signed int b7 : 2;
                signed int b8 : 2;
                signed int b9 : 2;
                signed int b10: 2;
                signed int b11: 2;
                signed int b12: 2;
```

```
                signed int b13: 2;
                signed int b14: 2;
                signed int b15: 2;
                signed int b16: 2;
        } ;
        struct lnibles{
                signed int n1 : 4;
                signed int n2 : 4;
                signed int n3 : 4;
                signed int n4 : 4;
                signed int n5 : 4;
                signed int n6 : 4;
                signed int n7 : 4;
                signed int n8 : 4;
        };
        struct lbytes{
                signed int n1 : 8;
                signed int n2 : 8;
                signed int n3 : 8;
                signed int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                signed  char  data[2048];
        } in ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                signed long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{ i = 0;
        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",
```

```
        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                printf("What is the file you want counted:\n");
                scanf("%s",newfile);
                printf("\nWORKING!!!!\n");

                f=fopen(newfile,"rb");/* Open selected data file */
                        n=0;
                        z=0;
                        do
                        {
                                afb2[z]=z;/*This puts numbers into the array for comparing*/
                                counter2[z]=0;/*This puts zeros in the array to set the count to
 zero*/
                                z++;
                        }while(z<=15);z=0;/*Puts numbers up to 15 in the array*/
                        do
                        {
                                afb1[n]=n;/*This puts numbers into the array for comparing*/
                                counter1[n]=0;/*This puts zeros in the array to set the count to
 zero*/
                                n++;
                        }while(n<=255);n=0;/*Puts numbers up to 255 in the array*/
```

```
while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
{       nf = nf + 1;/*Frame counter*/
                j=0;
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                {
                        do
                        {
                                if(in.data[j]==afb1[n])/*Check if the numbers
match*/
                                {
                                        counter1[n]+=1;/*counter if they
match*/
                                        break;/*Breaks once counted*/
                                }

                        }while(n<=255);n=0;
                }
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                {
                        do
                        {
                                if(in.data[j]==(-1*afb1[n]))
                                {
                                        negative1[n]+=1;
                                        printf("%l count
%l",in.data[j],negative1[n]);
                                        hello=getchar();
                                        break;
                                }
                                n++;
                        }while(n<=255);n=0;
                }
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data into nibles*/
                        {       out.nible[i]  .n3 = in.nible[j]  .n2;
                                out.nible[i]  .n2 = in.nible[j]  .n1;
                                out.nible[i]  .n1 = in.nible[j+1].n2;
                                out.nible[i+1].n3 = in.nible[j+1].n1;
                                out.nible[i+1].n2 = in.nible[j+2].n2;
                                out.nible[i+1].n1 = in.nible[j+2].n1;
                                i=i+2;
                        }
                        for (j=0; j<nwpf[fi]*nbpw[fi]/12; j++)/*Loop through the
frame of 12 bit words*/
                        { z=0;
                                do
                                {
```

```
                                        if(out.nible[j].n1==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }
                                        if(out.nible[j].n2==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }
                                        if(out.nible[j].n3==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }

                                z++;
                                }while(z<=15);z=0;
                        }
                        z=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/12; j++)/*Loop through the
frame of 12 bit words*/
                        { z=0;
                                do
                                {
                                        if(out.nible[j].n1==(afb2[z]*(-1)))
                                        {
                                                negative2[z]+=1;
                                                printf("%l count
%l",out.nible[j].n1,negative2[n]);
                                                hello=getchar();
                                        }
                                        if(out.nible[j].n2==(afb2[z]*(-1)))
                                        {
                                                negative2[z]+=1;
                                                printf("%l count
%l",out.nible[j].n2,negative2[n]);
                                                hello=getchar();
                                        }
                                        if(out.nible[j].n3==(afb2[z]*(-1)))
                                        {
                                                negative2[z]+=1;
                                                printf("%l count
%l",out.nible[j].n2,negative2[n]);
                                                hello=getchar();
                                        }
                                z++;
                                }while(z<=15);z=0;
                        } printf(".");
```

```
            }
        printf("\n%s DONE!!!!\n",newfile);
        printf("\nOutput file name for the two hex #: ");
        scanf("%s",newfile1);
        printf("\nOutput file name for the one hex #: ");
        scanf("%s",newfile2);
        lp2=fopen(newfile2,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp2,"%s\n",filename[fi]);/*Prints file name to the output file*/
        lp1=fopen(newfile1,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp1,"%s\n",filename[fi]);/*Prints file name to the output file*/
        n=0;
        do
        {
                fprintf(lp1,"%lx\tCount\t%ld\n",afb1[n],(long)counter1[n]);/*Printing to the
output file*/
                n++;
        }while(n<=255);n=0;
        z=0;
        do
        {
                fprintf(lp1,"%lx\tCount\t%ld\n",(afb1[n]*(-1)),(long)negative1[n]);
                n++;
        }while(n<=255);n=0;
        z=0;

        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long)counter2[z]);/*Printing to the
output file*/
                z++;
        }while(z<=15);z=0;
        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",(afb1[n]*(-1)),(long)negative2[n]);
                z++;
        }while(z<=15);z=0;

        printf ("\nSuccessful EndOfFile reached %ld Minor Frames.\n\n",(long) nf);/* Cleanup,
close-up, and start over */

        fclose(f);
        fclose(lp1);
        fclose(lp2);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Prints menu of files*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

NEW TWO

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int b=0,i, j, z=0,x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles],
sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int nf=0,fsp[numfiles],
nbps[numfiles],afb1[257],counter1[257],afb2[257],counter2[257],afb3[257],counter3[257];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30], newfile[40];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
```

```
                unsigned int b14: 2;
                unsigned int b15: 2;
                unsigned int b16: 2;
        } ;
        struct lnibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
                unsigned int n3 : 4;
                unsigned int n4 : 4;
                unsigned int n5 : 4;
                unsigned int n6 : 4;
                unsigned int n7 : 4;
                unsigned int n8 : 4;
        };
        struct lbytes{
                unsigned int n1 : 8;
                unsigned int n2 : 8;
                unsigned int n3 : 8;
                unsigned int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
unsigned long int temp;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{ i = 0;
        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",
```

```
        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                        newfile[0]='e';
                        newfile[1]=':';
                        b=0;
                         do
                         {
                                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
 name so it can read it off a cd*/
                                b++;
                         }while(b<=10);b=0;
                 printf("\nWORKING!!!!\n");

                 f=fopen(newfile,"rb");/* Open selected data file */
                        n=0;
                        z=0;
                        do
                        {
                                afb2[z]=z;/*This puts numbers into the array for comparing*/
                                counter2[z]=0;/*This puts zeros in the array to set the count to
 zero*/
                                z++;
                         }while(z<=15);z=0;/*Puts numbers up to 15 in the array*/
```

```
                do
                {
                        afb1[n]=n;/*This puts numbers into the array for comparing*/
                        counter1[n]=0;/*This puts zeros in the array to set the count to
zero*/
                        n++;
                }while(n<=255);n=0;/*Puts numbers up to 255 in the array*/
        while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
        {       nf = nf + 1;/*Frame counter*/
                        j=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                        {
                                do
                                {
                                        if(in.data[j]==afb1[n])/*Check if the numbers
match*/
                                        {
                                                counter1[n]+=1;/*counter if they
match*/
                                                break;/*Breaks once counted*/
                                        }
                                        n++;
                                }while(n<=255);n=0;
                        }
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data into nibles*/
                        {       out.nible[i]  .n3 = in.nible[j]  .n2;
                                out.nible[i]  .n2 = in.nible[j]  .n1;
                                out.nible[i]  .n1 = in.nible[j+1].n2;
                                out.nible[i+1].n3 = in.nible[j+1].n1;
                                out.nible[i+1].n2 = in.nible[j+2].n2;
                                out.nible[i+1].n1 = in.nible[j+2].n1;
                                i=i+2;
                        }
                        for (j=0; j<nwpf[fi]*nbpw[fi]/12; j++)/*Loop through the
frame of 12 bit words*/
                        { z=0;
                                do
                                {
                                        if(out.nible[j].n1==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }
                                        if(out.nible[j].n2==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
```

```
                                                }
                                                if(out.nible[j].n3==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                                {
                                                        counter2[z]+=1;
                                                }
                                        z++;
                                        }while(z<=15);z=0;
                                }

                }
        printf("\n%s DONE!!!!\n",filename[fi]);
        printf("\nOutput file name for the two hex #: ");
        scanf("%s",newfile1);
        printf("\nOutput file name for the one hex #: ");
        scanf("%s",newfile2);
        lp2=fopen(newfile2,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp2,"%s\n",filename[fi]);/*Prints file name to the output file*/
        lp1=fopen(newfile1,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp1,"%s\n",filename[fi]);/*Prints file name to the output file*/
        n=0;
        do
                {
                        if(counter1[n]<counter1[n-1])
                                { printf("...");
                                        temp=counter1[n-1];
                                        counter1[n-1]=counter1[n];
                                        counter1[n]=temp;
                                        afb1[257]=afb1[n-1];
                                        afb1[n-1]=afb1[n];
                                        afb1[n]=afb1[257];
                                }
                        n++;
                }while(n<=255);n=0;
        do
        {
                fprintf(lp1,"%lx\tCount\t%ld\n",afb1[n],(long)counter1[n]);/*Printing to the
output file*/
                n++;
        }while(n<=255);n=0;
        z=0;
        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long)counter2[z]);/*Printing to the
output file*/
                z++;
        }while(z<=15);z=0;
        printf ("\nSuccessful EndOfFile reached %ld Minor Frames.\n\n",(long) nf);/* Cleanup,
close-up, and start over */
```

```
        fclose(f);
        fclose(lp1);
        fclose(lp2);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Prints menu of files*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

READ1

```
/*
*-------------------------------------------------------------
*
*       Name:                                   Display_Data.c
*       Creation Date:                  09-Sep-97
*       Description:                    Display Data from Telemetry Sample Data Sets
*       Author:                                 Vernon Diekmann, TYBRIN
*       Target Machine/OS/Compiler:     Intel Pentium/Windows 95/NT/MS Visual C++ 5.0
*-------------------------------------------------------------
*
*       Revision History
*
*       Initials Date           Description
*       -------- ----           -----------
*       VD              09/09/97        Initial version
*       VD              10/10/97        Added 12 and 16 bit
*       VD              01/15/98        Added 24 bit and read byte records
*       VD              01/28/98        Input the file name for the data.dat file.
*
-------------------------------------------------------------
*/
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int i, j, nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int nf=0, nfrms[numfiles], wow[lengfiln];
unsigned long int fsp[numfiles], nbps[numfiles];
char response, excel[32];
char hello;
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
```

```
};
struct bytes{
        unsigned int n1 : 8;
};
struct ltbits
{
        unsigned int b1 : 2;
        unsigned int b2 : 2;
        unsigned int b3 : 2;
        unsigned int b4 : 2;
        unsigned int b5 : 2;
        unsigned int b6 : 2;
        unsigned int b7 : 2;
        unsigned int b8 : 2;
        unsigned int b9 : 2;
        unsigned int b10: 2;
        unsigned int b11: 2;
        unsigned int b12: 2;
        unsigned int b13: 2;
        unsigned int b14: 2;
        unsigned int b15: 2;
        unsigned int b16: 2;
} ;
struct lnibles{
        unsigned int n1 : 4;
        unsigned int n2 : 4;
        unsigned int n3 : 4;
        unsigned int n4 : 4;
        unsigned int n5 : 4;
        unsigned int n6 : 4;
        unsigned int n7 : 4;
        unsigned int n8 : 4;
};
struct lbytes{
        unsigned int n1 : 8;
        unsigned int n2 : 8;
        unsigned int n3 : 8;
        unsigned int n4 : 8;
};
union
{
        struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes    byte[2048];
        unsigned  char  data[2048];
} in ;
union
{
```

```
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp;

void main()
{

        for (i = 0; i < pltln; i++) plot[i] = ' ';/* Initialize array */

        i = 0;

        fin=fopen("cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %d\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                printf ("  Enter\nF to print each frame or \nW to plot a single word: \n");
```

```
response = getchar();

if (response == 'w' || response == 'W')
{
        printf ("  Enter the\n Word, Subframe, Interval to display: \n");
        result = scanf("%d%d%d",&word,&subframe,&interval);
        hello = getchar();

        printf ("  Enter 'c' to continue or type the path and file name for the data of
the word selected: \n");
        result = scanf("%s",&excel);
}
f=fopen(filename[fi],"rb");/* Open selected data file */

result = strlen(&excel[0]);

if(result > 2)
        fo=fopen(&excel[0],"w");/* Open option data output file for excel */

while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
{       nf = nf + 1;
        if (response == 'f' || response == 'F')/* Display each minor frame */
        {       printf ("\nFrame=%d\n",nf);
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                {
                        printf("%02x",in.data[j]);
                }
                printf("\n");
                printf ("Enter to continue <ctrl>C to quit");
                hello=getchar();
        }
        else if (response == 'w' || response == 'W') /* Display selected data
parameter */
        {
                if (nbpw[fi]==8)
                {
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=1)/* Unravel packed 8
bit data */
                        {
                                out.byte[i].n1  = in.byte[j].n1;
                        }
                }
                if (nbpw[fi]==10)
                {
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=5)/* Unravel packed 10
bit data */
                        {
```

```
                    out.tbit[i].b5   = in.tbit[j].b4;
                    out.tbit[i].b4   = in.tbit[j].b3;
                    out.tbit[i].b3   = in.tbit[j].b2;
                    out.tbit[i].b2   = in.tbit[j].b1;
                    out.tbit[i].b1   = in.tbit[j+1].b4;
                    out.tbit[i+1].b5 = in.tbit[j+1].b3;
                    out.tbit[i+1].b4 = in.tbit[j+1].b2;
                    out.tbit[i+1].b3 = in.tbit[j+1].b1;
                    out.tbit[i+1].b2 = in.tbit[j+2].b4;
                    out.tbit[i+1].b1 = in.tbit[j+2].b3;
                    out.tbit[i+2].b5 = in.tbit[j+2].b2;
                    out.tbit[i+2].b4 = in.tbit[j+2].b1;
                    out.tbit[i+2].b3 = in.tbit[j+3].b4;
                    out.tbit[i+2].b2 = in.tbit[j+3].b3;
                    out.tbit[i+2].b1 = in.tbit[j+3].b2;
                    out.tbit[i+3].b5 = in.tbit[j+3].b1;
                    out.tbit[i+3].b4 = in.tbit[j+4].b4;
                    out.tbit[i+3].b3 = in.tbit[j+4].b3;
                    out.tbit[i+3].b2 = in.tbit[j+4].b2;
                    out.tbit[i+3].b1 = in.tbit[j+4].b1;
                    i=i+4;
            }
        }

        if (nbpw[fi]==12)
        {
            i=0;
            for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data */
            {       out.nible[i]  .n3 = in.nible[j]  .n2;
                    out.nible[i]  .n2 = in.nible[j]  .n1;
                    out.nible[i]  .n1 = in.nible[j+1].n2;
                    out.nible[i+1].n3 = in.nible[j+1].n1;
                    out.nible[i+1].n2 = in.nible[j+2].n2;
                    out.nible[i+1].n1 = in.nible[j+2].n1;
                    i=i+2;
            }
        }
        if (nbpw[fi]==16)
        {
            i=0;
            for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=2)/* Unravel packed 16
bit data */
            {       out.byte[i].n2 = in.byte[j]  .n1;
                    out.byte[i].n1 = in.byte[j+1].n1;
                    i=i+1;
            }
        }
        if (nbpw[fi]==24)
```

```
                {
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 24
bit data */
                        {       out.byte[i].n3 = in.byte[j]  .n1;
                                out.byte[i].n2 = in.byte[j+1].n1;
                                out.byte[i].n1 = in.byte[j+2].n1;
                                i=i+1;
                        }
                }
                if ((sfidw[fi] >= 0) & ( interval != 0) & (subframe >= 0))/* Handle
sub frame data */
                {       sfid = out.data[sfidw[fi]-1];
                        sfid = sfid >> rs;
                        for (j=0; j<(sfl[fi]-sff[fi]+1); j+=interval)
                        {       if ((subframe+j) == sfid)
                                {       data = (unsigned)out.data[word-1]/pltln;
                                        data = (unsigned)out.data[word-1]-
data*pltln;
                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2)fprintf (fo,"%d\t%d\n",nf,
out.data[word-1]);
                                }
                        }
                }
                else if ((subframe == -1) & ( interval != 0))/* Handle super
commutated data */
                {       for (j=0; j<=nwpf[fi]; j+=interval)

                                {       data = (unsigned)out.data[word-1+j]/pltln;
                                        data = (unsigned)out.data[word-1+j]-
data*pltln;
                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2 )fprintf(fo,"%d\t%d\n",nf,
out.data[word-1+j]);
                                }

                }
                else /* Just plain old data */
                {       data = (unsigned)out.data[word-1]/pltln;
                        data = (unsigned)out.data[word-1]-data*pltln;
```

```
                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2)fprintf (fo,"%d\t%d\n",nf, out.data[word-1]);
                                }
                        }
                        else
                        {       /* Do nothing */
                        }
                }
        printf ("Successful EndOfFile reached %d Minor Frames.\n", nf);/* Cleanup, close-up,
and start over */
        fclose(f);

        if (result > 2)fclose(fo);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

TEN GREA

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
unsigned int b=0,i,x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int nfrms[numfiles], wow[lengfiln];
unsigned long int z=0, j, l, n, nf=0,fsp[numfiles], nbps[numfiles],afb1[4096],counter1[4096];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30],newfile[40];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
                unsigned int b15: 2;
```

```
            unsigned int b16: 2;
        } ;
        struct lnibles{
            unsigned int n1 : 4;
            unsigned int n2 : 4;
            unsigned int n3 : 4;
            unsigned int n4 : 4;
            unsigned int n5 : 4;
            unsigned int n6 : 4;
            unsigned int n7 : 4;
            unsigned int n8 : 4;
        };
        struct lbytes{
            unsigned int n1 : 8;
            unsigned int n2 : 8;
            unsigned int n3 : 8;
            unsigned int n4 : 8;
        };
        union
        {
            struct tbits   tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
            struct nibles  nible[2048];
            struct bytes   byte[2048];
            unsigned  char  data[2048];
        } in ;
        union
        {
            struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
            struct lnibles   nible[4096];
            struct lbytes    byte[4096];
            unsigned long int data[4096];
        } out ;
        unsigned long int temp;
        FILE *f, *fin, *fo, *lp1, *lp2, *lp3, *nice;
        unsigned long int n=0;
void main()
{ unsigned long int l=0;
        i = 0;
        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
```

```
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                        newfile[0]='e';
                        newfile[1]=':';
                        b=0;
                         do
                         {
                                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off a cd*/
                                b++;
                         }while(b<=10);b=0;
                printf("\nWORKING!!!!\n");

                f=fopen(newfile,"rb");/* Open selected data file */
                        n=0;
                        z=0;
                do
                        {
                                afb1[n]=n;
                                counter1[n]=0;
                                n++;
                        }while(n<=1024);n=0;
                while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
                {       nf = nf + 1;/*Frame counter*/
                                i=0;
```

```
                    for (j=0;j<nwpf[fi]*nbpw[fi]/8;j=j+5)/* Unravel packed 10 bit data
*/
                                {
                                        out.tbit[i].b5   = in.tbit[j].b4;
                                        out.tbit[i].b4   = in.tbit[j].b3;
                                        out.tbit[i].b3   = in.tbit[j].b2;
                                        out.tbit[i].b2   = in.tbit[j].b1;
                                        out.tbit[i].b1   = in.tbit[j+1].b4;
                                        out.tbit[i+1].b5 = in.tbit[j+1].b3;
                                        out.tbit[i+1].b4 = in.tbit[j+1].b2;
                                        out.tbit[i+1].b3 = in.tbit[j+1].b1;
                                        out.tbit[i+1].b2 = in.tbit[j+2].b4;
                                        out.tbit[i+1].b1 = in.tbit[j+2].b3;
                                        out.tbit[i+2].b5 = in.tbit[j+2].b2;
                                        out.tbit[i+2].b4 = in.tbit[j+2].b1;
                                        out.tbit[i+2].b3 = in.tbit[j+3].b4;
                                        out.tbit[i+2].b2 = in.tbit[j+3].b3;
                                        out.tbit[i+2].b1 = in.tbit[j+3].b2;
                                        out.tbit[i+3].b5 = in.tbit[j+3].b1;
                                        out.tbit[i+3].b4 = in.tbit[j+4].b4;
                                        out.tbit[i+3].b3 = in.tbit[j+4].b3;
                                        out.tbit[i+3].b2 = in.tbit[j+4].b2;
                                        out.tbit[i+3].b1 = in.tbit[j+4].b1;
                                        i=i+4;
                                }
                                        for(n=0;n<nwpf[fi]*nbpw[fi]/10; n++)
                                        {
                                                for(l=0;l<=1024;l=l+1)
                                                {
                                                        if(afb1[l]==out.data[n])
                                                                {
                                                                        counter1[l]++;
                                                                        break;
                                                                }
                                                }
                                        }n=0;
                                }
                printf("\n\nDone\n\n");
                printf("What is the name of the output file: ");
                scanf("%s",newfile2);
                nice=fopen(newfile2,"w");
                fprintf(nice,"%s\n",filename[fi]);
                z=0;
                do
                        {
                                fprintf(nice,"%lx\tCount\t%ld\n",(long)afb1[z],(long)counter1[z]);
                                z=z+1;
                        }while(z<=1024);z=0;
        printf("\nFile Saved at %s\n\n",newfile2);
```

```
        fclose(nice);
        fclose(f);
        fclose(lp1);
        fclose(lp2);
        nf = 0;
        i=0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Prints menu of files*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

TWOONE

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int i, j, z=0,x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int nf=0,fsp[numfiles],
nbps[numfiles],afb1[256],counter1[256],afb2[256],counter2[256],afb3[256],counter3[256];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
```

```
                unsigned int b15: 2;
                unsigned int b16: 2;
        } ;
        struct lnibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
                unsigned int n3 : 4;
                unsigned int n4 : 4;
                unsigned int n5 : 4;
                unsigned int n6 : 4;
                unsigned int n7 : 4;
                unsigned int n8 : 4;
        };
        struct lbytes{
                unsigned int n1 : 8;
                unsigned int n2 : 8;
                unsigned int n3 : 8;
                unsigned int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{ i = 0;
        fin=fopen("cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
```

```
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                printf("HIT RETURN TO GO ON!!!");
                hello=getchar();
                printf("\nWORKING!!!\n");
                f=fopen(&filename[fi][0],"rb");/* Open selected data file */
                        n=0;
                        z=0;
                        do
                        {
                                afb2[z]=z;/*This puts numbers into the array for comparing*/
                                counter2[z]=0;/*This puts zeros in the array to set the count to
zero*/
                                z++;
                        }while(z<=15);z=0;/*Puts numbers up to 15 in the array*/
                        do
                        {
                                afb1[n]=n;/*This puts numbers into the array for comparing*/
                                counter1[n]=0;/*This puts zeros in the array to set the count to
zero*/
                                n++;
                        }while(n<=255);n=0;/*Puts numbers up to 255 in the array*/
                while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
                {       nf = nf + 1;/*Frame counter*/
                                j=0;
                                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                                {
```

```
                        do
                        {
                                if(in.data[j]==afb1[n])/*Check if the numbers
match*/
                                {
                                        counter1[n]+=1;/*counter if they
match*/
                                        break;/*Breaks once counted*/
                                }
                                n++;
                        }while(n<=255);n=0;
                }
                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data into nibles*/
                        {       out.nible[i]  .n3 = in.nible[j]  .n2;
                                out.nible[i]  .n2 = in.nible[j]  .n1;
                                out.nible[i]  .n1 = in.nible[j+1].n2;
                                out.nible[i+1].n3 = in.nible[j+1].n1;
                                out.nible[i+1].n2 = in.nible[j+2].n2;
                                out.nible[i+1].n1 = in.nible[j+2].n1;
                                i=i+2;
                        }
                        for (j=0; j<nwpf[fi]*nbpw[fi]/12; j++)/*Loop through the
frame of 12 bit words*/
                        { z=0;
                                do
                                {
                                        if(out.nible[j].n1==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }
                                        if(out.nible[j].n2==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }
                                        if(out.nible[j].n3==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                                        {
                                                counter2[z]+=1;
                                        }
                                z++;
                                }while(z<=15);z=0;
                        }

                }
        printf("\n%s DONE!!!!\n",filename[fi]);
```

```
        printf("\nOutput file name for the two hex #: ");
        scanf("%s",newfile1);
        printf("\nOutput file name for the one hex #: ");
        scanf("%s",newfile2);
        lp2=fopen(newfile2,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp2,"%s\n",filename[fi]);/*Prints file name to the output file*/
        lp1=fopen(newfile1,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp1,"%s\n",filename[fi]);/*Prints file name to the output file*/
        n=0;
        do
        {
                fprintf(lp1,"%lx\tCount\t%ld\n",afb1[n],(long)counter1[n]);/*Printing to the
output file*/
                n++;
        }while(n<=255);n=0;
        z=0;
        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long)counter2[z]);/*Printing to the
output file*/
                z++;
        }while(z<=15);z=0;
        printf ("\nSuccessful EndOfFile reached %ld Minor Frames.\n\n",(long) nf);/* Cleanup,
close-up, and start over */

        fclose(f);
        fclose(lp1);
        fclose(lp2);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Prints menu of files*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

TWOONECD

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int b=0,i, j, z=0,x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles],
sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int nf=0,fsp[numfiles],
nbps[numfiles],afb1[256],counter1[256],afb2[256],counter2[256],afb3[256],counter3[256];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30], newfile[40];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
```

```
                unsigned int b14: 2;
                unsigned int b15: 2;
                unsigned int b16: 2;
        } ;
        struct lnibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
                unsigned int n3 : 4;
                unsigned int n4 : 4;
                unsigned int n5 : 4;
                unsigned int n6 : 4;
                unsigned int n7 : 4;
                unsigned int n8 : 4;
        };
        struct lbytes{
                unsigned int n1 : 8;
                unsigned int n2 : 8;
                unsigned int n3 : 8;
                unsigned int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{ i = 0;
        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
```

```
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;

                        newfile[0]='e';
                        newfile[1]=':';
                        b=0;
                         do
                         {
                                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file
name so it can read it off a cd*/
                                b++;
                         }while(b<=10);b=0;
                printf("\nWORKING!!!!\n");

                f=fopen(newfile,"rb");/* Open selected data file */
                        n=0;
                        z=0;
                        do
                        {
                                afb2[z]=z;/*This puts numbers into the array for comparing*/
                                counter2[z]=0;/*This puts zeros in the array to set the count to
zero*/
                                z++;
                        }while(z<=15);z=0;/*Puts numbers up to 15 in the array*/
                        do
                        {
                                afb1[n]=n;/*This puts numbers into the array for comparing*/
```

```
                counter1[n]=0;/*This puts zeros in the array to set the count to
zero*/
                n++;
            }while(n<=255);n=0;/*Puts numbers up to 255 in the array*/
        while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
        {       nf = nf + 1;/*Frame counter*/
                j=0;
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j++)
                {
                    do
                    {
                        if(in.data[j]==afb1[n])/*Check if the numbers
match*/
                        {
                            counter1[n]+=1;/*counter if they
match*/
                            break;/*Breaks once counted*/
                        }
                        n++;
                    }while(n<=255);n=0;
                }
                i=0;
                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data into nibles*/
                {       out.nible[i]  .n3 = in.nible[j]  .n2;
                        out.nible[i]  .n2 = in.nible[j]  .n1;
                        out.nible[i]  .n1 = in.nible[j+1].n2;
                        out.nible[i+1].n3 = in.nible[j+1].n1;
                        out.nible[i+1].n2 = in.nible[j+2].n2;
                        out.nible[i+1].n1 = in.nible[j+2].n1;
                        i=i+2;
                }
                for (j=0; j<nwpf[fi]*nbpw[fi]/12; j++)/*Loop through the
frame of 12 bit words*/
                { z=0;
                    do
                    {
                        if(out.nible[j].n1==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                        {
                            counter2[z]+=1;
                        }
                        if(out.nible[j].n2==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
                        {
                            counter2[z]+=1;
                        }
                        if(out.nible[j].n3==afb2[z])/*Checks if the
numbers match up and then counts it, goes bit by bit*/
```

```
                                        {
                                                counter2[z]+=1;
                                        }
                                z++;
                                }while(z<=15);z=0;
                        }

                }
        printf("\n%s DONE!!!!\n",filename[fi]);
        printf("\nOutput file name for the two hex #: ");
        scanf("%s",newfile1);
        printf("\nOutput file name for the one hex #: ");
        scanf("%s",newfile2);
        lp2=fopen(newfile2,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp2,"%s\n",filename[fi]);/*Prints file name to the output file*/
        lp1=fopen(newfile1,"w");/*Opens a file to put the numbers and the count into*/
        fprintf(lp1,"%s\n",filename[fi]);/*Prints file name to the output file*/
        n=0;
        do
        {
                fprintf(lp1,"%lx\tCount\t%ld\n",afb1[n],(long)counter1[n]);/*Printing to the
output file*/
                n++;
        }while(n<=255);n=0;
        z=0;
        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long)counter2[z]);/*Printing to the
output file*/
                z++;
        }while(z<=15);z=0;
        printf ("\nSuccessful EndOfFile reached %ld Minor Frames.\n\n",(long) nf),/* Cleanup,
close-up, and start over */

        fclose(f);
        fclose(lp1);
        fclose(lp2);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }/*Prints menu of files*/
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

WORD

```
/*
*-------------------------------------------------------------------
*
*       Name:                                   Display_Data.c
*       Creation Date:                  09-Sep-97
*       Description:                    Display Data from Telemetry Sample Data Sets
*       Author:                                 Vernon Diekmann, TYBRIN
*       Target Machine/OS/Compiler:     Intel Pentium/Windows 95/NT/MS Visual C++ 5.0
*-------------------------------------------------------------------
*
*       Revision History
*
*       Initials Date           Description
*       -------- ----           -----------
*       VD              09/09/97        Initial version
*       VD              10/10/97        Added 12 and 16 bit
*       VD              01/15/98        Added 24 bit and read byte records
*       VD              01/28/98        Input the file name for the data.dat file.
*
-------------------------------------------------------------------
*/
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int i, j, x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int z=0,nf=0,fsp[numfiles],
nbps[numfiles],afb1[256],counter1[256],afb2[4096],counter2[4096],afb3[256],counter3[256];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
```

```
        unsigned int n2 : 4;
};
struct bytes{
        unsigned int n1 : 8;
};
struct ltbits
{
        unsigned int b1 : 2;
        unsigned int b2 : 2;
        unsigned int b3 : 2;
        unsigned int b4 : 2;
        unsigned int b5 : 2;
        unsigned int b6 : 2;
        unsigned int b7 : 2;
        unsigned int b8 : 2;
        unsigned int b9 : 2;
        unsigned int b10: 2;
        unsigned int b11: 2;
        unsigned int b12: 2;
        unsigned int b13: 2;
        unsigned int b14: 2;
        unsigned int b15: 2;
        unsigned int b16: 2;
} ;
struct lnibles{
        unsigned int n1 : 4;
        unsigned int n2 : 4;
        unsigned int n3 : 4;
        unsigned int n4 : 4;
        unsigned int n5 : 4;
        unsigned int n6 : 4;
        unsigned int n7 : 4;
        unsigned int n8 : 4;
};
struct lbytes{
        unsigned int n1 : 8;
        unsigned int n2 : 8;
        unsigned int n3 : 8;
        unsigned int n4 : 8;
};
union
{
        struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
        struct nibles  nible[2048];
        struct bytes    byte[2048];
        unsigned  char  data[2048];
} in ;
union
```

```
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{

        for (i = 0; i < pltln; i++) plot[i] = ' ';/* Initialize array */

        i = 0;

        fin=fopen("cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
```

```
printf ("  Enter\nF to print each frame or \nW to plot a single word: \n");
response = getchar();

if (response == 'w' || response == 'W')
{
        printf ("  Enter the\n Word, Subframe, Interval to display: \n");
        result = scanf("%d%d%d",&word,&subframe,&interval);
        hello = getchar();

        printf ("  Enter 'c' to continue or type the path and file name for the data of
the word selected: \n");
        result = scanf("%s",&excel);
}
f=fopen(&filename[fi][0],"rb");/* Open selected data file */

result = strlen(&excel[0]);

if(result > 2)
        fo=fopen(&excel[0],"w");/* Open option data output file for excel */
        j=0;
         z=0;
        do
        {
                afb2[z]=z;
                counter2[z]=0;
                z++;
        }while(z<=4095);z=0;
 printf("\nHello\n");
while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
{       nf = nf + 1;    printf("%ld\n",nf);
if (response == 'f' || response == 'F')/* Display each minor frame */
        {       /**printf ("\nFrame=%d\n",nf);**/

                        i=0;
                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
 bit data */

                        {       out.nible[i]  .n3 = in.nible[j]  .n2;
                                out.nible[i]  .n2 = in.nible[j]  .n1;
                                out.nible[i]  .n1 = in.nible[j+1].n2;
                                out.nible[i+1].n3 = in.nible[j+1].n1;
                                out.nible[i+1].n2 = in.nible[j+2].n2;
                                out.nible[i+1].n1 = in.nible[j+2].n1;
                                i=i+2;
                        }
                        for(i=0; i<nwpf[fi]*nbpw[fi]/12; i++)
                        { z=0;
                                do
                                {
                                        if(out.data[i]==afb2[z])
```

```
                                        {
                                                counter2[z]+=1;

                                                break;
                                        }
                                        z++;
                                }while(z<=4095);z=0;
                        }
                }
                else if (response == 'w' || response == 'W') /* Display selected data
parameter */
                {
                        if (nbpw[fi]==8)
                        {
                                i=0;
                                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=1)/* Unravel packed 8
bit data */
                                {
                                        out.byte[i].n1  = in.byte[j].n1;
                                }
                        }
                        if (nbpw[fi]==10)
                        {
                                i=0;
                                for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=5)/* Unravel packed 10
bit data */
                                {
                                        out.tbit[i].b5   = in.tbit[j].b4;
                                        out.tbit[i].b4   = in.tbit[j].b3;
                                        out.tbit[i].b3   = in.tbit[j].b2;
                                        out.tbit[i].b2   = in.tbit[j].b1;
                                        out.tbit[i].b1   = in.tbit[j+1].b4;
                                        out.tbit[i+1].b5 = in.tbit[j+1].b3;
                                        out.tbit[i+1].b4 = in.tbit[j+1].b2;
                                        out.tbit[i+1].b3 = in.tbit[j+1].b1;
                                        out.tbit[i+1].b2 = in.tbit[j+2].b4;
                                        out.tbit[i+1].b1 = in.tbit[j+2].b3;
                                        out.tbit[i+2].b5 = in.tbit[j+2].b2;
                                        out.tbit[i+2].b4 = in.tbit[j+2].b1;
                                        out.tbit[i+2].b3 = in.tbit[j+3].b4;
                                        out.tbit[i+2].b2 = in.tbit[j+3].b3;
                                        out.tbit[i+2].b1 = in.tbit[j+3].b2;
                                        out.tbit[i+3].b5 = in.tbit[j+3].b1;
                                        out.tbit[i+3].b4 = in.tbit[j+4].b4;
                                        out.tbit[i+3].b3 = in.tbit[j+4].b3;
                                        out.tbit[i+3].b2 = in.tbit[j+4].b2;
                                        out.tbit[i+3].b1 = in.tbit[j+4].b1;
                                        i=i+4;
                                }
```

```
                    }

                    if (nbpw[fi]==12)
                    {
                            i=0;
                            for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
bit data */
                            {       out.nible[i]  .n3 = in.nible[j]  .n2;
                                    out.nible[i]  .n2 = in.nible[j]  .n1;
                                    out.nible[i]  .n1 = in.nible[j+1].n2;
                                    out.nible[i+1].n3 = in.nible[j+1].n1;
                                    out.nible[i+1].n2 = in.nible[j+2].n2;
                                    out.nible[i+1].n1 = in.nible[j+2].n1;
                                    i=i+2;
                            }
                    }
                    if (nbpw[fi]==16)
                    {
                            i=0;
                            for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=2)/* Unravel packed 16
bit data */
                            {       out.byte[i].n2 = in.byte[j]  .n1;
                                    out.byte[i].n1 = in.byte[j+1].n1;
                                    i=i+1;
                            }
                    }
                    if (nbpw[fi]==24)
                    {
                            i=0;
                            for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 24
bit data */
                            {       out.byte[i].n3 = in.byte[j]  .n1;
                                    out.byte[i].n2 = in.byte[j+1].n1;
                                    out.byte[i].n1 = in.byte[j+2].n1;
                                    i=i+1;
                            }
                    }
                    if ((sfidw[fi] >= 0) & ( interval != 0) & (subframe >= 0))/* Handle
sub frame data */
                    {       sfid = out.data[sfidw[fi]-1];
                            sfid = sfid >> rs;
                            for (j=0; j<(sfl[fi]-sff[fi]+1); j+=interval)
                            {       if ((subframe+j) == sfid)
                                    {       data = (unsigned)out.data[word-1]/pltln;
                                            data = (unsigned)out.data[word-1]-
data*pltln;
                                            plot[data] = '.';
                                            plot[data+1] = '\0';
                                            printf ("%d%s\n",nf, plot);
```

```
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2)fprintf (fo,"%d\t%d\n",nf,
out.data[word-1]);
                                }
                        }
                }
                else if ((subframe == -1) & ( interval != 0))/* Handle super
commutated data */
                {       for (j=0; j<=nwpf[fi]; j+=interval)

                                {       data = (unsigned)out.data[word-1+j]/pltln;
                                        data = (unsigned)out.data[word-1+j]-
data*pltln;

                                        plot[data] = '.';
                                        plot[data+1] = '\0';
                                        printf ("%d%s\n",nf, plot);
                                        plot[data] = ' ';
                                        plot[data+1] = ' ';
                                        if (result > 2 )fprintf(fo,"%d\t%d\n",nf,
out.data[word-1+j]);
                                }

                }
                else /* Just plain old data */
                {       data = (unsigned)out.data[word-1]/pltln;
                        data = (unsigned)out.data[word-1]-data*pltln;
                        plot[data] = '.';
                        plot[data+1] = '\0';
                        printf ("%d%s\n",nf, plot);
                        plot[data] = ' ';
                        plot[data+1] = ' ';
                        if (result > 2)fprintf (fo,"%d\t%d\n",nf, out.data[word-1]);
                }
        }
        else
        {       /* Do nothing */
        }
}


printf("\nOutput file name for the word data: ");
scanf("%s",newfile2);
lp2=fopen(newfile2,"w");
fprintf(lp2,"%s\n",filename[fi]);

z=0;
do
{
```

```
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long) counter2[z]);
                z++;
        }while(z<=4095);z=0;
        printf ("Successful EndOfFile reached %ld Minor Frames.\n", (long)nf);/* Cleanup,
close-up, and start over */
        fclose(f);
        fclose(lp2);
        if (result > 2)fclose(fo);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }
        printf ("\nEnter the file Number: \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

WORDCD

```
#include <stdio.h>
#include <string.h>
#define numfiles 10
#define lengfiln 40
#define pltln 125
char filename[numfiles][lengfiln], ofilename[lengfiln], ifilename[lengfiln], outdat[lengfiln],
plot[pltln+1];
char fsll[numfiles][lengfiln];
int result, status, subframe, fi, a;
unsigned int data,word, interval, rs, sfid;
unsigned  int i, j, x=0,nwpf[numfiles], nbpw[numfiles], sfidw[numfiles], sfido[numfiles],
        sff[numfiles], sfl[numfiles], numfile, fsl[numfiles], nb;
unsigned int b=0,n=0, nfrms[numfiles], wow[lengfiln];
unsigned long int z=0,nf=0,fsp[numfiles],
nbps[numfiles],afb1[256],counter1[256],afb2[4096],counter2[4096],afb3[256],counter3[256];
char response, excel[32];
char hello,newfile1[30],newfile2[30],newfile3[30], newfile[40];
        struct tbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
        } ;
        struct nibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
        };
        struct bytes{
                unsigned int n1 : 8;
        };
        struct ltbits
        {
                unsigned int b1 : 2;
                unsigned int b2 : 2;
                unsigned int b3 : 2;
                unsigned int b4 : 2;
                unsigned int b5 : 2;
                unsigned int b6 : 2;
                unsigned int b7 : 2;
                unsigned int b8 : 2;
                unsigned int b9 : 2;
                unsigned int b10: 2;
                unsigned int b11: 2;
                unsigned int b12: 2;
                unsigned int b13: 2;
                unsigned int b14: 2;
```

```
                unsigned int b15: 2;
                unsigned int b16: 2;
        } ;
        struct lnibles{
                unsigned int n1 : 4;
                unsigned int n2 : 4;
                unsigned int n3 : 4;
                unsigned int n4 : 4;
                unsigned int n5 : 4;
                unsigned int n6 : 4;
                unsigned int n7 : 4;
                unsigned int n8 : 4;
        };
        struct lbytes{
                unsigned int n1 : 8;
                unsigned int n2 : 8;
                unsigned int n3 : 8;
                unsigned int n4 : 8;
        };
        union
        {
                struct tbits    tbit[2048];/* Max allowable Class II bytes per frame 16384/8 =
2048*/
                struct nibles  nible[2048];
                struct bytes    byte[2048];
                unsigned  char  data[2048];
        } in ;
        union
        {
                struct ltbits    tbit[4096];/* Max allowable Class II wpf = 16384/4 = 4096*/
                struct lnibles   nible[4096];
                struct lbytes    byte[4096];
                unsigned long int data[4096];
        } out ;
FILE *f, *fin, *fo, *lp1, *lp2, *lp3;

void main()
{       i = 0;

        fin=fopen("e:cinput.dat","rb");  /* Open file and read data description information */
        for     (status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]);
                status != EOF;
                status = fscanf(fin,"%s %ld %d %d %ld %d %d %d %d %d %s %ld\n",

        &filename[i],&fsp[i],&nwpf[i],&nbpw[i],&nbps[i],&sfidw[i],&sfido[i],&sff[i],&sfl[i],&
fsl[i],&fsll[i],&nfrms[i]))
```

```
        {       if (i < numfiles-1) i = i +1;
        }
        numfile = i ;
        fclose (fin);

        for (i=0; i < numfile; i++){ printf("%d---->%s\n",i,filename[i]);}/* Display Menu */
        i=0;    fi = 0;  result = 0;
        printf ("  Enter the file Number : \n");/* Make Selection */
        for     (result=scanf("%d",&fi); fi != -1; result=scanf("%d",&fi))/* Main loop executed
once for each file selection */
        {       hello = getchar(); nf = 0;/* First display and collect option information */
                printf("\nFile = %s\t\tFrame Sync Pattern = %lx\nNumber of Words Per Frame =
%d\t\tNumber of Bits Per Word = %d\n",
                filename[fi],fsp[fi],nwpf[fi],nbpw[fi]);
                printf("Sub Frame ID Word = %d\t\t\tSub Frame ID Offset = %d\nSub Frame
First = %d\t\t\tSub Frame Last = %d\n",
                sfidw[fi],sfido[fi],sff[fi],sfl[fi]);
                printf("Number of Bits Per Second = %ld\tTotal Number of Frames = %ld\n",
                nbps[fi],(long) nfrms[fi]);
                printf("Frame Sync Length       = %d\t\tSync Location       = %s\n\n",
                fsl[fi],fsll[fi]);
                nb=0;i=1;
                while ((sfl[fi]-sff[fi])>i)       {nb=nb+1;i=2*i;}
                rs = nbpw[fi]-sfido[fi]-nb+1;
                newfile[0]='e';
                        newfile[1]=':';
                        b=0;
                         do
                         {
                                newfile[2+b]=filename[fi][b];/*This puts 'e:' in front of the file so
it can read off a cd*/
                                b++;
                         }while(b<=10);b=0;

                printf("\nWORKING!!!!\n");
                 f=fopen(newfile,"rb");/* Open selected data file */
                        j=0;
                         z=0;
                        do
                        {
                                afb2[z]=z;
                                counter2[z]=0;
                                z++;
                        }while(z<=4095);z=0;
                while (fread(in.data,(nwpf[fi]*nbpw[fi]/8),1,f))/* Read data records until EOF */
                {       nf = nf + 1;
                                i=0;
                                        for (j=0; j<nwpf[fi]*nbpw[fi]/8; j+=3)/* Unravel packed 12
 bit data */
```

```
                                {       out.nible[i]  .n3 = in.nible[j]  .n2;
                                        out.nible[i]  .n2 = in.nible[j]  .n1;
                                        out.nible[i]  .n1 = in.nible[j+1].n2;
                                        out.nible[i+1].n3 = in.nible[j+1].n1;
                                        out.nible[i+1].n2 = in.nible[j+2].n2;
                                        out.nible[i+1].n1 = in.nible[j+2].n1;
                                        i=i+2;
                                }
                                for(i=0; i<nwpf[fi]*nbpw[fi]/12; i++)
                                { z=0;
                                        do
                                        {
                                                if(out.data[i]==afb2[z])
                                                {
                                                        counter2[z]+=1;

                                                        break;
                                                }
                                                z++;
                                        }while(z<=4095);z=0;
                                }
                        }
        printf("\n\nFILE %s DONE\n\n",filename[fi]);
        printf("\nOutput path and file name for the word data: ");
        scanf("%s",newfile2);
        lp2=fopen(newfile2,"w");
        fprintf(lp2,"%s\n",filename[fi]);

        z=0;
        do
        {
                fprintf(lp2,"%lx\tCount\t%ld\n",afb2[z],(long) counter2[z]);
                z++;
        }while(z<=4095);z=0;
        printf ("Successful EndOfFile reached %ld Minor Frames.\n", (long)nf);/* Cleanup,
close-up, and start over */
        fclose(f);
        fclose(lp2);
        if (result > 2)fclose(fo);
        nf = 0;
        for (i=0; i < numfile; i++){ printf("%d %s\n",i,filename[i]); }
        printf ("  Enter the file Number : \n");
        printf ("  <ctrl> c to end: \n");
        }
} /* End Main */
```

Shakti Davis wrote code to implement the channel simulations using the error files supplied by TYBRIN. The compression algorithm (lzss is from the Mark Nelson data compression book). These computer programs and instructions are listed below:

Documentation for BERA Simulations:

To run a BERA simulation, open MATLAB and type "bera_sim" at the command prompt. This should bring up a dialog box in which you can specify the transmit data file, the receive data file and whether you want the data to be compressed (with lzss) before the errors are injected. The last parameter, channel coding isn't implemented yet. To get the simulation started, you can either hit OK to accept the default parameters, or you can change any of the first three parameters. For the filenames, I would recommend giving the full path name (e.g. g:\sdsdata\sds008c.dat instead of just sds008c.dat) to insure that the simulation uses the correct file. Notice that the files can be read from the CD ROM drive, so you won't have to copy the files to the harddrive. Please be patient with the simulation because it may take several minutes to complete. If, however, the simulation is running and no harddrive activity occurs for several minutes, then the program may have frozen. In this case, I would suggest hitting CTRL-ALT-DEL and ending the MATLAB session.

The simulation is composed of several files acting on the data files. First, the "BERA_sim.m" MATLAB file collects the simulation parameters from the user and determines whether the data file should be compressed. If the user indicated "yes" for compression, then the C-file lzss.c is used to compress the file. This is possible by compiling the C file as a MEX function. To compile the lzss.c file as a MEX file, do the following:

1. Open MATLAB
2. From the prompt, change the working directory path to D:\stats\simulink simulation\lzss which is where the C compression code resides

```
>> cd 'D:\stats\simulink simulation\lzss'
```

3. Compile the compression routine as a MEX file:

```
>>mex -DMATLAB -output 'd:\stats\simulink simulation\lzss_c' main-c.c lzss.c
bitio.c errhand.c
```

this creates a file called lzss_c.dll in the d:\stats\simulink simulation\ directory that can be called from MATLAB

4. Use this function in a MATLAB script as a normal function call

```
>>lzss_c('SDS001c.dat','SDS001c.cmp');
```

So the BERA_sim file compiles the input data file using this lzss_c MEX function and moves on to the next step which is to process the BERA file. The processing is also done in a C program that was compiled as a MEX function. The C program is called procBERA.c and is located at d:\stats\simulink simulation\. To compile this program as a MEX file, do the following:

1. Open MATLAB
2. From the prompt, change the working directory path to D:\stats\simulink simulation which is where the procBERA C source code resides

```
>> cd 'D:\stats\simulink simulation\lzss'
```

3. Compile the compression routine as a MEX file:

```
>>mex -DMATLAB -output 'd:\stats\simulink simulation\procbera' procbera.c
```

this creates a file called procbera.dll in the d:\stats\simulink simulation\ directory that can be called from MATLAB

4. Use this function in a MATLAB script as a normal function call
>>procBERA('d:\stats\flight2bera\F02f_02_M12_O28.bin', 'd:\stats\SDS001c.dat', 'd:\stats\rcvddata.bin')

This procBERA.dll function is called whether the compression is done or not, so this MEX function must exist in order to run the BERA simulation. The procBERA C code takes as inputs the name of the BERA error file, the name of the data input file, and the name of the data output file. Each of these files is opened and a replica of the data input file is copied to the data output file (this is the easiest way since there are so few errors relative to the number of correct bits). So originally the input data is equal to the output data. Next, the error file is read one entry at a time, where each error entry consists of the following 6-byte structure:

XOR word : bits 47-32
Event Byte : bits 31-24
24 bit counter : bits 23-0

For each error, an absolute position of that error is specified by the 24-bit counter, so the program goes to that position in the output data file. The XOR word contains a 1 at each error location, so once the output data file is correctly positioned, its data at that position (2-bytes), can be read, XOR-ed with the XOR word of the error entry, and written back to the data output file. This process is repeated for each error event in the BERA error file, or until the end of the data file is reached. The BERA files cover such a long transmit signal that the input data file is usually the limiting factor (e.g. the position specified by the error events are larger than the last position in the input data file). The event byte of the error structure is not used. While the error file is being processed, statistics about the file are collected. The three statistics which are returned to the MATLAB program are the bit-error-rate, the number of bits in error, and the total number of bits processed. To access these return values from the MATLAB call, simply call procBERA as follows:

```
[BER,numErrs,numBits]=procBERA(BERfile,infile,outfile);
```

and the bit error rate will be returned in the BER variable, the number of bits in error will be returned in the numErrs variable, and the total number of bits processed will bw returned in the numBits variable. If the simulation was specified to use the compression, then the data file that is created by the procBERA function must be decompressed using the lzss_e MEX function (compiled the same way as the lzss_c MEX function but instead of main-c.c use main-e.c). After the file is decompressed, it can be compared with the original data file and the true bit-error rate can be determined.

BERA simulation program:

```
%Simulation using BERA files

% prompt for data file names
Prompts = { 'Enter the BERA filename:',...
      'Enter the filename of the original data:',...
      'Enter the filename for the output data:',...
      'Do you want to compress the data first? (Yes/No)',...
      'Do you want to use channel coding? (Yes/No)'};
def = {'e:\stats\flight2bera\F02f_02_M12_028.bin',...
      'e:\stats\SDS001c.dat',...
      'e:\stats\rcvddata.dat',...
      'No',...
      'No'};
lineNo=1;
title = 'Simulation Configuration';
Answer = inputdlg(Prompts,title,lineNo,def);

if (strcmpi(char(Answer(4)),'yes'))
   % make the compressed data filename equal to the original
   % data filename with the extension ".cmp" and save it to the
   % stats directory
   index=findstr(char(Answer(2)),'.');
   if (index)
      infile=char(Answer(2));
      infile=infile(1:index-1);
      infile=strcat(infile,'.cmp');
   else
      infile=char(Answer(2));
      infile=strcat(infile,'.cmp');
   end
   index=findstr(char(Answer(2)),'\');
   if (index)
      infile=strcat('e:\stats\',infile(index(end)+1:end));
   else
      infile=strcat('e:\stats\',infile);
   end


   % compress the input data file
   lzss_c(char(Answer(2)),infile);

   % make the compressed output filename equal to the output
   % filename with the extension ".cmp"
   index=findstr(char(Answer(3)),'.');
   if (index)
      outfile=char(Answer(3));
      outfile=outfile(1:index-1);
      outfile=strcat(outfile,'.cmp');
   else
      outfile=char(Answer(3));
      outfile=strcat(outfile,'.cmp');
   end
else
   infile = char(Answer(2));
   outfile = char(Answer(3));
end;
```

```
% process the error file
[BER,actualErrs,actualBits]=procBERA(char(Answer(1)),infile,outfile);

if (strcmpi(char(Answer(4)),'yes'))
   %decompress the output file and calculate the BER
   lzss_e(outfile,char(Answer(3)));
   [BER,actualErrs,actualBits]=checkBER(char(Answer(1)),char(Answer(2)));
end

% sumarize the simulation
fprintf(1,'Scenario:\n');
fprintf(1,'\tInput file:\t\t\t%s\n',char(Answer(1)));
fprintf(1,'\tOutput file:\t\t\t%s\n',char(Answer(2)));
fprintf(1,'\tData Compressed?\t\t%s\n',char(Answer(4)));
fprintf(1,'\tChannel Coding?\t\t\t%s\n',char(Answer(5)));
fprintf(1,'Statistics:\n');
fprintf(1,'\tBit Error Rate:\t\t\t%e\n',BER);
fprintf(1,'\tNumber of Bits Processed:\t%d\n',actualBits);
fprintf(1,'\tNumber of Errors:\t\t%d\n',actualErrs);
```

PROCBERA program (the error file was supplied by SyntheSys Research):

```
ifndef _BA25READ_H_
#define _BA25READ_H_
/*
 *-------------------------------------------*
 * Global Constant Definitions
 *-------------------------------------------*
 */
#define ROLL_PERIOD_25  0x01000000
#define ROLL_BYTES_25   (ROLL_PERIOD_25 << 1)
#define ROLL_MASK_25    (ROLL_BYTES_25 -1)
#define WORD_SIZE_25    16
/*
 *-------------------------------------------*
 * Global Macro Definitions
 *-------------------------------------------*
 */
#define HW25_EVENT_BYTE(Ptr)   (*(((unsigned char *)(Ptr))+3))
/*
 *-------------------------------------------*
 * Type Definitions
 *-------------------------------------------*
 */
/*
 * BitAlyzer 25 Event Package Structure
 *
 * Note: Xor is declared as a short instead of a bit field
 *       so that sizeof(Sample25) is correctly evaluated to
 *       6 bytes.
 */


 typedef struct {

  unsigned long Clock    :24;
  unsigned long Marker   :1;
  unsigned long Pattern  :1;
  unsigned long Resync   :1;
  unsigned long Squelch  :1;
  unsigned long MarkerPosition :4;
  unsigned short Xor ;
  } Sample25 ;

  typedef struct {
          unsigned long data     :16;
  }Word
```

```
#endif
/* EOF */


/* BA25READ.C */
/*
================================================================================
====*/
/* The objective of this program is to show how a BitAlyzer 25 Error File
   could be read using a simple program written in C.  To understand this
   program the user must be familiar with the BitAlyzer error file format.

   This program reads an error file and computes the total number of errors
   and the total error free interval.

   For further information on writing programs to access error file data
   or error file format specifications please contact:  SyntheSys Research,
   Inc. at 650-364-1853

        Modified 11/18/99 -- S.D.
              Now this program locates the errors and actually XORs the error bits
              with the true data.  The resulting file can then be compared to the
              original data file for error analysis.

              Syntax: procBERA BERAfile binarydatafile outputfile

   Notes from BYU:
   ---------------
   The BitAlyzer stores it's data in 6 byte chunks.

  XOR word      :  bits 47-32
  Event Byte    :  bits 31-24
  24 bit counter :  bits 23-0

  Because the data was written by a PC, and we were doing our work on Unix
  boxes, we had a little-endian big-endian problem (i.e. the byte
  addressing in the file is opposite between the PC and Unix boxes.)  To
  overcome this problem I just read each byte out individually, and
  processed them the way I wanted.  I hope that the documentation in the
  code is clear.  Another reason to open the data in Byte chucks is since
  the data is 6 Bytes long, my attempts to try and read a 6 Byte block
  were actually executed as a read of 6 Bytes, then skip to the next word,
  i.e. two bytes later.  So I was losing data.  I could tell I was having
  problems because I was comparing it to a known file.

  24 bit POSITION
  (bit pos: 0..23)
```

These 24 bits hold a count (in 16 bit words) of user clocks. For example, an error in the 100th serial bit of a serial stream would have POSITION count of 6 indicating an error presence in the 7th 16 bit word. To find which exact bit of the word was in error, one would look at which bit in the XOR word was a logical "1" (mismatch). In this example, it would be the 4th bit. It is guaranteed that at least one event will occur per the 16,777,218 counts of the 24 bit counter to allow a software implementation of a larger counter.
(The guaranteed event promised to detect the clock rollover is that the 24 bit counter reads 2, and the XOR word is all zeros. So all I did to detect clock roll over was to see if one the current clock value was less than the previous.)

bits EVENT
(bit pos: 24..31)

Four of the eight bits in this byte are modifiers to the event type. These modifiers are orthogonal in that one event could have several modifiers. For example, an event could have errors AND be a used as a marker AND be a RESYNC location. The other four bits in the event byte are used to indicate the exact bit position in which the marker event occurred. THe marker position bits are only valid when the Marker event bit is set to 1. The modifiers are defined as follows:

b0 (bit pos: 24)	M-Marker (Rear Panel)
b1 (bit pos: 25)	P-Pattern Cycle Marker (Internal)
b2 (bit pos: 26)	R-Start Resynchronization Event
b3 (bit pos: 27)	S-Begin Error Squelch Event
b4-b7 (bit pos: 28-31)	Bit position of the Marker within this event word.

Marker: This is an internal pole once every second (for Edward's Air Force base). We didn't worry about this event.

Pattern: The BitAlyzer has acquired the pattern, and is now synched up. NOTE: The PATTERN event might only happen once in a file, Usually at the beginning of a file.

Resync: The BitAlyzer25 has reached a threshold of errors where it no longer is able to hold pattern, and now is in the process of trying to acquire pattern.

Squelch: To many errors have occurred and have filled the BitAlyzer internal buffer. The BitAlyzer empties the buffer, and then resumes detecting and writing errors to the file. During the squelch, errors are NOT recorded. The buffer is 2 k words, and is controlled by software.

Okay, now here comes some iffy stuff. When I asked them how to detect when the events were over, they told be the following. They said that for the Resync or Squelch to end, the ZERO event had to occur. The zero event was defined to be that the XOR word was all zeros, and that the EVENT Byte had to be all zeros. I used this in my code, but when I would compare it to known data I could never match up the results. So I used the fact that they said that no errors were recorded during a Squelch event, and some trial and error to decide for myself that the way to detect that the events are over if bits 24-27 are all zero, i.e. the marker events are zero. I did this and presto, my data matched the output their expensive code said that it should. I still have a small discrepancy on the number of bits lost in a squelch event, but the correct number of squelch events is calculated. I'm not sure if the discrepancy comes from the fact that I window the data or not. It's been 6 months since I've worked on the problem, and my memory is rusty on the details. The real trick to figuring out the event structure was to compare it to a known file.

The man to contact at Synthesis Research is a guy called Bert Carner (650) 364-1853, Bert_Carner@synthesysresearch.com. Tell him that you're working with Edward's Air Force Base, and he'll probably be helpful.

*/

```
#include <stdio.h>
#include <stdlib.h>
#include "ba25read.h"

#ifdef MATLAB
        #include "D:\MATLABR11\extern\include\mex.h"
#endif // ifdef MATLAB

/*
 * Declare local routines.
 */
void ProcessErrorFile( FILE *fin, FILE *fdata, FILE *fout, double*, double*, double*) ;

/*
  How the ErrsPerByte[] array is used to find the number of 1's:
  ErrsPerByte[BinaryNumberIndex]=No. of 1's in the BinaryNumberIndex
  ErrsPerByte[5]-->ErrsPerByte[b0101]--> two 1's = 2;
  */

unsigned char ErrsPerByte[] = {
  0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
```

```
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8} ;


/*
 Routine Definitions
 */
#ifndef MATLAB

 void main( int argc, char ** argv )
{
 FILE * errorFile ;
 FILE * dataFile ;
 FILE * outFile ;
 char fileXfer[MAX_XFER_SIZE];

        int numBytes;
        int numCycles=0;
        int i=0;
        int actErrors=0;
        double BER;
        double ActualErrs;
        double TotalBits;

  /*
   Open the BA25 Error File and process the events.
   */
  if ( argv[1] )
  {

                errorFile = fopen( argv[1], "rb") ;
                dataFile = fopen( argv[2], "rb");
                outFile = fopen( argv[3], "wb+");

                while ( !feof(dataFile) )
                {
                        // copy the data file to the ouput file
                        numBytes = fread(fileXfer,1,MAX_XFER_SIZE,dataFile);
                        fwrite(fileXfer,1,numBytes,outFile);
                        numCycles++;
```

```
            }

            rewind(dataFile);
            rewind(outFile);

            if ( errorFile && dataFile && outFile )
            {
                  ProcessErrorFile( errorFile, dataFile, outFile, &BER, &ActualErrs,
&TotalBits );
                  fclose( errorFile );
                  fclose( dataFile );
                  fclose( outFile );
            }
 }
 else
       printf("Syntax: procBERA BERAfile binarydatafile outputfile\n\n");

}
#endif


/*************************************************************************
 *      Function : void ProcessErrorFile( FILE * fin, FILE *fdata, FILE *fout,
                                                        double* BER, int*
ActualErrs, int* ActualBits )

 * Purpose/Descr.
 * Count the total number of bits processed and the total number of
 * errors recorded.  Also xor the errors with the original data file
 * to produce a new file with errors in the positions described by the
 * error file.

 * Input:
 * FILE * fin - pointer to the error event file to be processed
 * FILE * fdata -- pointer to the original error-free data file
 * FILE * fout  -- pointer to a replica of the data file.  This file will
                                    be modified with the bit errors
 * double * BER       -- double pointer to variable that will receive the actual Bit Error Rate
 *      int * ActualErrs -- int pointer to the variable that will receive the actual number of errors
 *      int * ActualBits -- int pointer to the variable that will receive the actual number of bits
 processed

 * Output: void

 * Return Codes: None

*************************************************************************/

void ProcessErrorFile( FILE *fin, FILE *fdata, FILE *fout,
```

```
                                        double* BER, double* ActualErrs, double* ActualBits )
{
 unsigned long          LastPosition = 0xffffffff;
 unsigned long          PositionCounter;
 unsigned short     EventByte;
 unsigned short     Xor;
 unsigned long          Efi;

 Sample25               ThisSample;
 double                 TotalBits=0;
 double                 TotalErrors=0;
 unsigned long          RollOvers=0;
 Word                   ThisWord;
 char                   fileComp1[MAX_XFER_SIZE];
 char                   fileComp2[MAX_XFER_SIZE];
 int                    numBytes;
 int                    numCycles=0;
 int                    i;

 rewind(fdata);
 rewind(fout);

      // read one record (16-bit word) from the error file
 while( fread( &ThisSample, sizeof(Sample25), 1, fin )) // 16-bit words
  {
              /*
             Extract the PositionCounter, Event, and Xor fields from the
             error event.
             */
              PositionCounter = ThisSample.Clock;
              EventByte = HW25_EVENT_BYTE( &ThisSample );
              Xor = ThisSample.Xor;


             /*
                    Check for Rollovers on the PostitionCounter
                    The guaranteed event promised to detect the clock rollover is that the
                    24 bit counter reads 2, and the XOR word is all zeros. So a rollover
                    occurred whenever the position counter is less than the last value of
                    the position counter.
             */
             if( (PositionCounter < LastPosition) && (LastPosition != 0xffffffff) )
                    RollOvers++;

             /*
             Process the Errors
             */
             if (Xor)
```

```
{
        /*
        Compute the error free interval.  "Efi" represents the error free
        interval as the number of error-free words.  Words are 16 bits long.

        If (PositionCounter - LastPosition) equals 1,
        then there are no error-free words in between the previous event
        and the current word.
        */
         if( LastPosition == 0xffffffff )
                // if LastPosition is 0xffffffff then this is the first error
                // so the error-free interval is equal to the PositionCounter
               Efi = PositionCounter ;
         else if( /*(PositionCounter > LastPosition) && */!RollOvers )
                // else if there haven't been any rollovers, then the error-free
                // interval is simply the difference between the two counters
                Efi = PositionCounter - LastPosition - 1L;
         else
                // otherwise at least one rollover has occured, so the math is
                // a little different
                Efi = RollOvers*ROLL_PERIOD_25 - (LastPosition -
PositionCounter) - 1L;

         // only keep track of how many rollovers occur between error events
         // so reset this counter every time an error is detected.
         RollOvers = 0;


        if (!feof(fdata))
        {
                // Seek to an of offset=Efi from the current position in the
                // data file.  That is where the next error(s) occurs.
                if (fseek(fdata,Efi*sizeof(Word),SEEK_CUR) ==0)
                {
                        ftell(fdata);
                        // Should read 2 bytes
                        numBytes = fread(&ThisWord,1,sizeof(Word),fdata);

                        if (numBytes)
                        {
                                // Flip the errored bits
                                ThisWord.byte0 ^= ((Xor >> 8) & 0xff);
                                ThisWord.byte1 ^= (Xor & 0xff);
                                // Go to the same position in the ouput file (received
data)
                                fseek(fout,Efi*sizeof(Word),SEEK_CUR);
                                // Write in the data with errors
                                fwrite(&ThisWord,1,numBytes,fout);
```

```
                                        // Add the number of errors in this word to total
error count
                                        if(numBytes>=1)
                                                TotalErrors += ErrsPerByte[ (Xor>>8) &
0xff ];
                                        if(numBytes>=2)
                                                TotalErrors += ErrsPerByte[ (Xor&0xff) ];

                                        /*
                                        Add the number of Error-Free Words (times 16
bits/word)
                                        plus the number of bits in the XOR field (16 bits)
                                        to the total number of bits processed.
                                        */
                                        TotalBits += (Efi * 16);
                                        TotalBits += 8*numBytes;
                                }
                        }
                        else
                                // else fseek couldn't locate this offset (EOF)
                                break;
                }
                else
                // if data file is shorter than the error file, then ignore the rest
                // of the error file.
                        break;
        }
        LastPosition = PositionCounter;
  }

#ifdef DEBUG
 /*
 Compare the output file to input file to determine the actual number of errors
 */
        rewind(fout);
        rewind(fdata);
        *ActualErrs=0;
        *ActualBits=0;

  while(!feof(fdata) && !feof(fout))
  {
                //read in a big chunk of the file each time
                ftell(fout);
                numBytes = fread(fileComp1,1,MAX_XFER_SIZE,fout);
                fread(fileComp2,1,MAX_XFER_SIZE,fdata);
                for(i=0; i<numBytes; i++)
                { // compare input and output data byte-by-byte
                        *ActualErrs += ErrsPerByte[(fileComp1[i] ^ fileComp2[i])&0xff];
                        *ActualBits += 8;
```

```
                }
 }
#else
 *ActualErrs = TotalErrors;
 fseek(fout,0L,SEEK_END);
 *ActualBits = ftell(fout)*8;
#endif

 *BER = (*ActualErrs)/(*ActualBits);

/*      printf( "Total Bits Processed = %.0lf\n",(float)(*ActualBits));
#ifdef DEBUG
        printf( "Total Errors Counted = %.0lf\n",TotalErrors);
#endif
        printf( "Total Actual Errors = %.0lf\n",(float)(*ActualErrs));
        printf( "Bit Error Rate = %6.4e\n",*BER);
*/
}


#ifdef MATLAB

// matlab gateway function makes it possible to call this function from matlab.
// to use this function in matlab, compile it on the matlab command line as follows:
//              1. change the working directory path to D:\stats\simulink simulation\
//                      >> cd 'D:\stats\simulink simulation\'
//              2. compile this program as a mex file:
//                      >> mex -DMATLAB -output procBERA procBERA.c
//    3. use this function in a MATLAB script as a normal function call
//                      >>procBERA('d:\stats\flight2bera\F02f_02_M12_O28.bin',
'd:\stats\SDS001c.dat', 'd:\stats\rcvddata.bin',BER,actualErrs,actualBits)

void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
{
        int buflen;
        char* errfile;
        char* datafile;
        char* outfile;
        FILE* errorFile;
        FILE* dataFile;
        FILE* outFile;
        char fileXfer[MAX_XFER_SIZE];
        int numBytes;
        int numCycles=0;
        double* BER;
        double* ActualErrs;
        double* TotalBits;

        /* Check for proper number of arguments. */
```

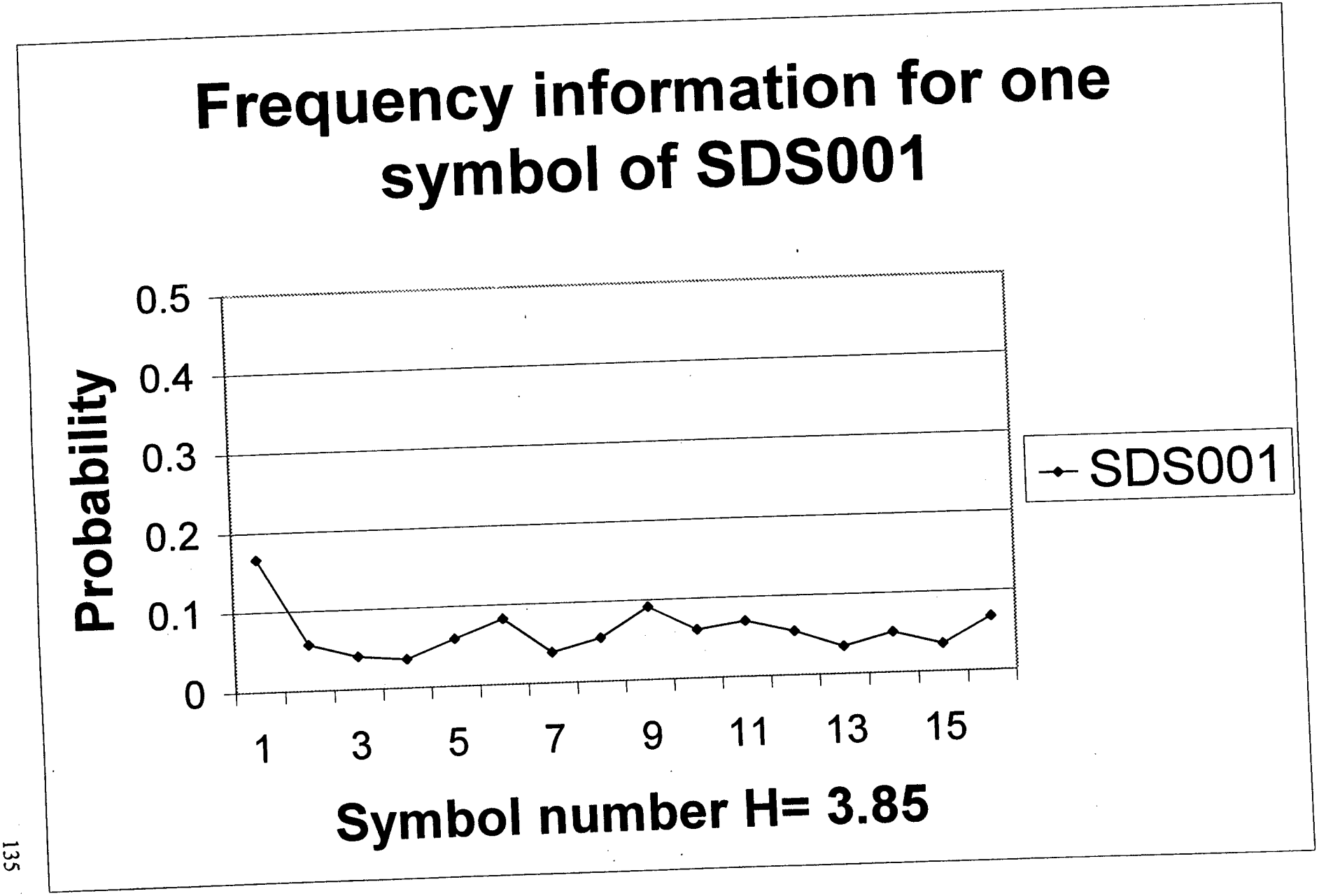
```
if(nrhs!=3)
{
        mexErrMsgTxt("Three inputs required.");
}
else if(nlhs>3) {
        mexErrMsgTxt("Too many output arguments");
}

/* The inputs must be strings */
if( !mxIsChar(prhs[0]) || !mxIsChar(prhs[1]) || !mxIsChar(prhs[2]))
{
        mexErrMsgTxt("Inputs must be strings.");
}

/* Get the length of the input string. */
buflen = (mxGetM(prhs[0]) * mxGetN(prhs[0])) + 1;
/* Allocate memory for input and output strings. */
errfile=mxCalloc(buflen, sizeof(char));
/* Assign pointers to each input and output. */
 mxGetString(prhs[0],errfile,buflen);

/* Get the length of the input string. */
buflen = (mxGetM(prhs[1]) * mxGetN(prhs[1])) + 1;
/* Allocate memory for input and output strings. */
datafile=mxCalloc(buflen, sizeof(char));
/* Assign pointers to each input and output. */
mxGetString(prhs[1],datafile,buflen);

/* Get the length of the input string. */
buflen = (mxGetM(prhs[2]) * mxGetN(prhs[2])) + 1;
/* Allocate memory for input and output strings. */
outfile=mxCalloc(buflen, sizeof(char));
/* Assign pointers to each input and output. */
mxGetString(prhs[2],outfile,buflen);

/* Create matrix for the return argument. */
plhs[0] = mxCreateDoubleMatrix(1,1, mxREAL);
plhs[1] = mxCreateDoubleMatrix(1,1, mxREAL);
plhs[2] = mxCreateDoubleMatrix(1,1, mxREAL);

BER = mxGetPr(plhs[0]);
ActualErrs = mxGetPr(plhs[1]);
TotalBits = mxGetPr(plhs[2]);

errorFile = fopen( errfile, "rb") ;
dataFile = fopen( datafile, "rb");
outFile = fopen( outfile, "wb+");

// Copy the data file to the output file
```

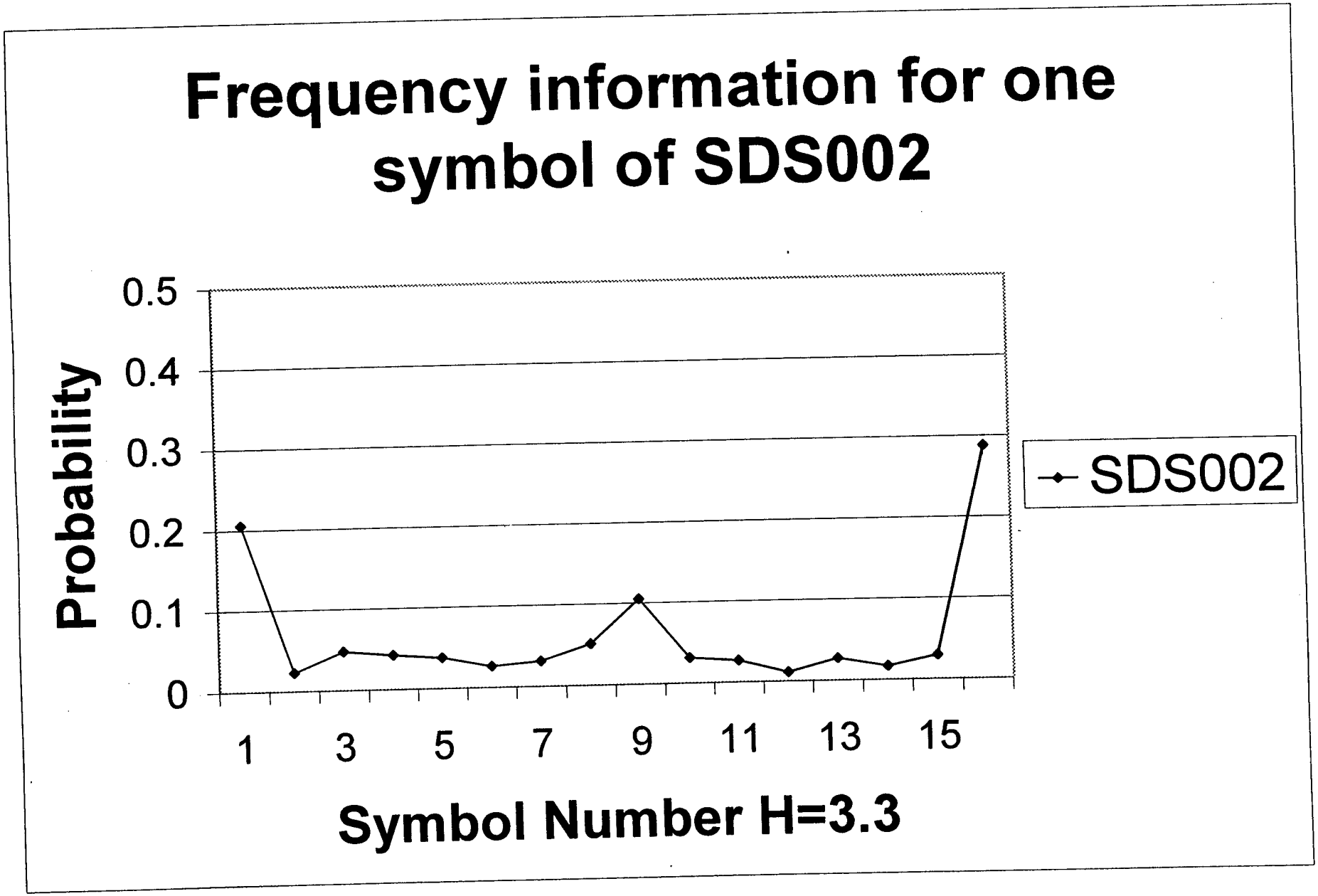
```
        while ( !feof(dataFile) )
        {
                numBytes = fread(fileXfer,1,MAX_XFER_SIZE,dataFile);
                fwrite(fileXfer,1,numBytes,outFile);
                numCycles++;
        }

        rewind(dataFile);
        rewind(outFile);

        if ( errorFile && dataFile && outFile )
        {
                ProcessErrorFile( errorFile, dataFile, outFile, BER, ActualErrs, TotalBits );
                fclose( errorFile );
                fclose( dataFile );
                fclose( outFile );
        }
        else
                printf("Syntax: procBERA\(BERAfile, binarydatafile, outputfile\)\n\n");
}


#endif
```

```
function numErrs=errsPerByte(byte)

if(nargin ~= 1)
   error('ErrsPerByte must have one argument: ErrsPerByte(Byte)\n')
else
   errors=[...
  0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4,...
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,...
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,...
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,...
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,...
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,...
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,...
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,...
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,...
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,...
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,...
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,...
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,...
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,...
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,...
  4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8];


   numErrs=errors(byte+1);
end

function [BER,numErrs,numBits]=checkErrorRate(infile, outfile)
%checkErrorRate

if(nargin ~= 2)
   error('checkErrorRate needs 2 arguments: checkErrorRate(infile,
outfile)\n');
else
   fin = fopen(infile,'rb');
   fout = fopen(outfile,'rb');
   numErrs=0;
   if((fin~=0) & (fout~=0))
      while (feof(fin) == 0)
         [input,numBytes]=fread(fin,hex2dec('1000000'),'uchar');
         output=fread(fout,numBytes,'uchar');
%          for(i=1:numBytes)
            numErrs = numErrs +
ErrsPerByte(bitand(bitxor(input(1),output(1)),hex2dec('ff')));
 %        end
      end
   end
end
```

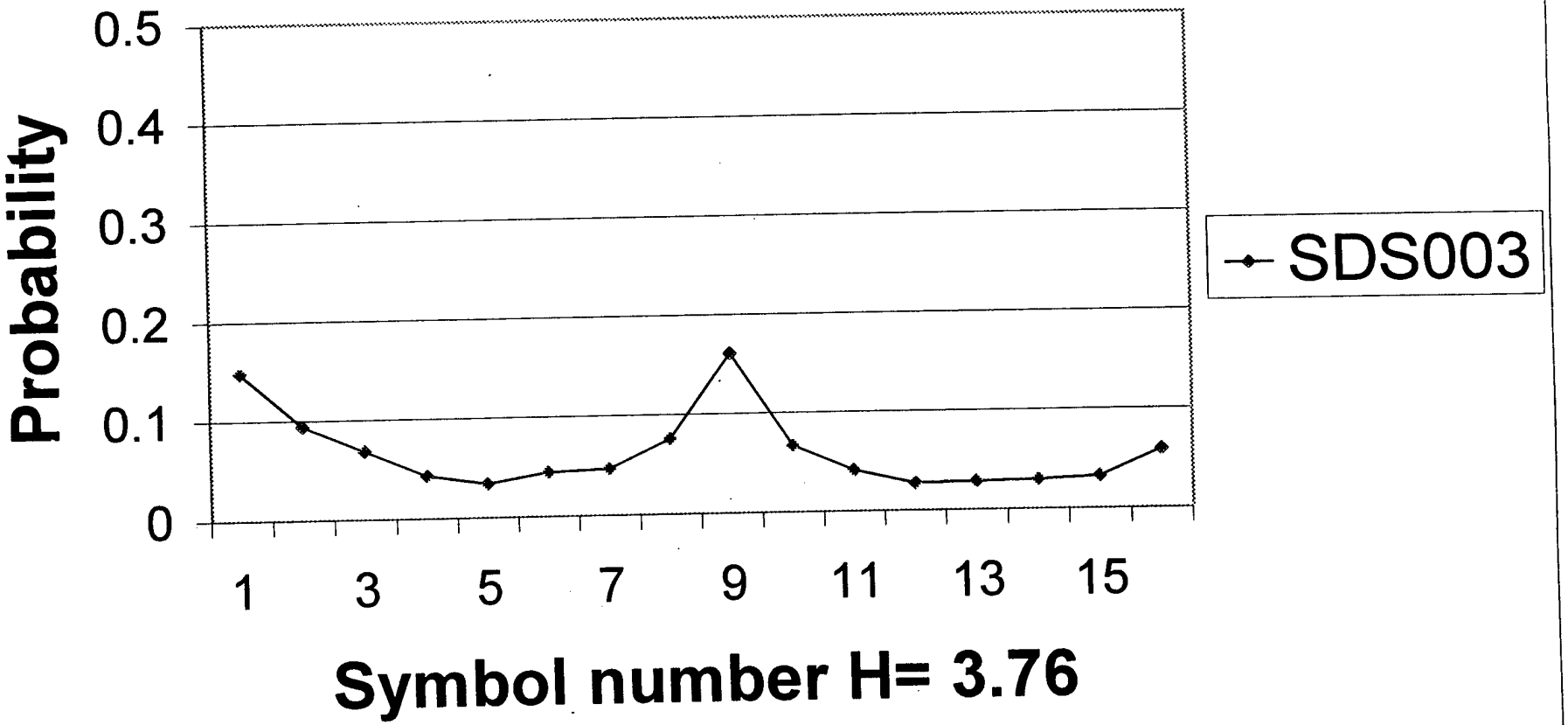
Data Statistics Appendix

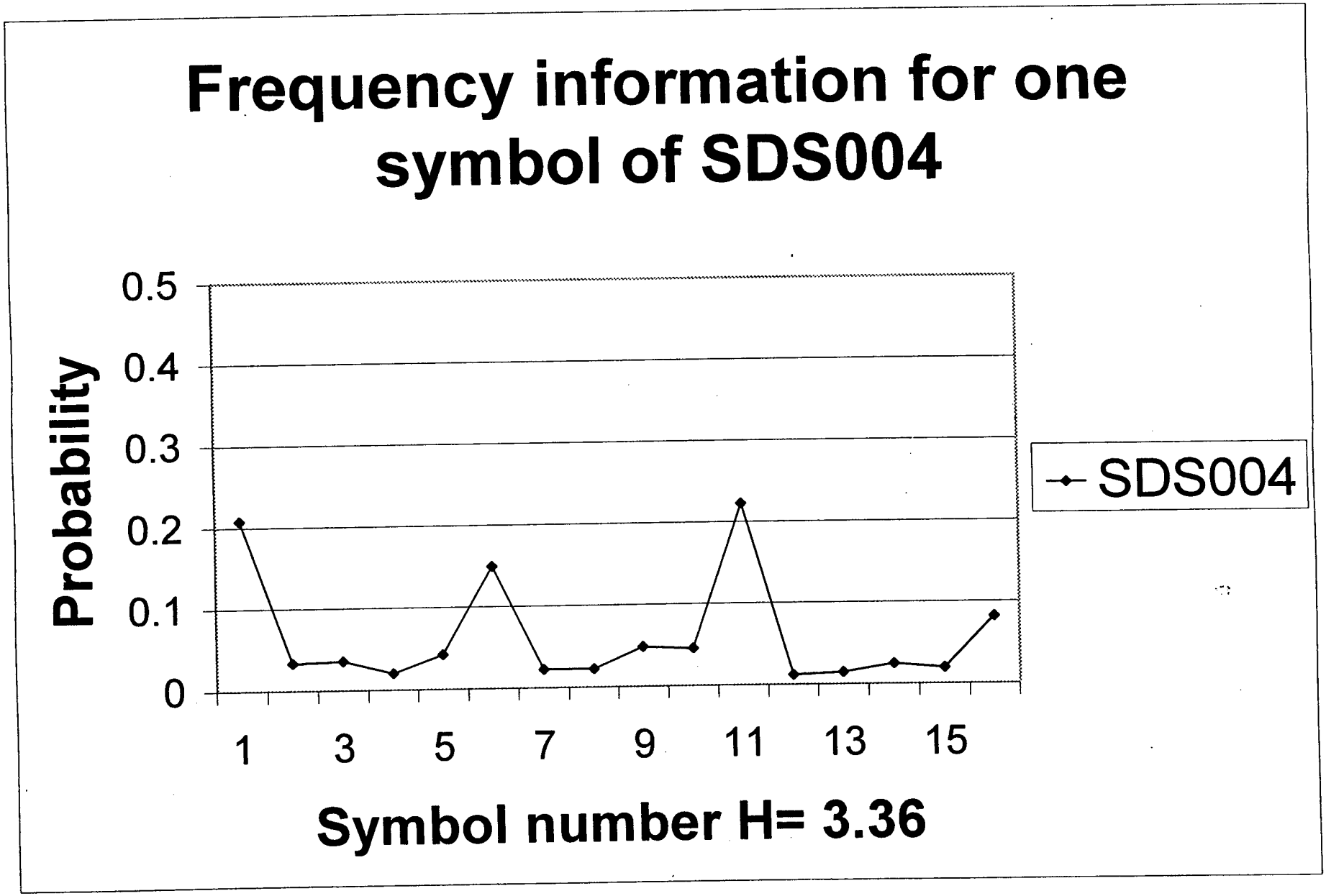
The following pages contain the statistical analysis for the probabilities and entropies for each of the data sets.


Frequency information for one symbol of SDS001
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS001
Symbol number H= 3.85

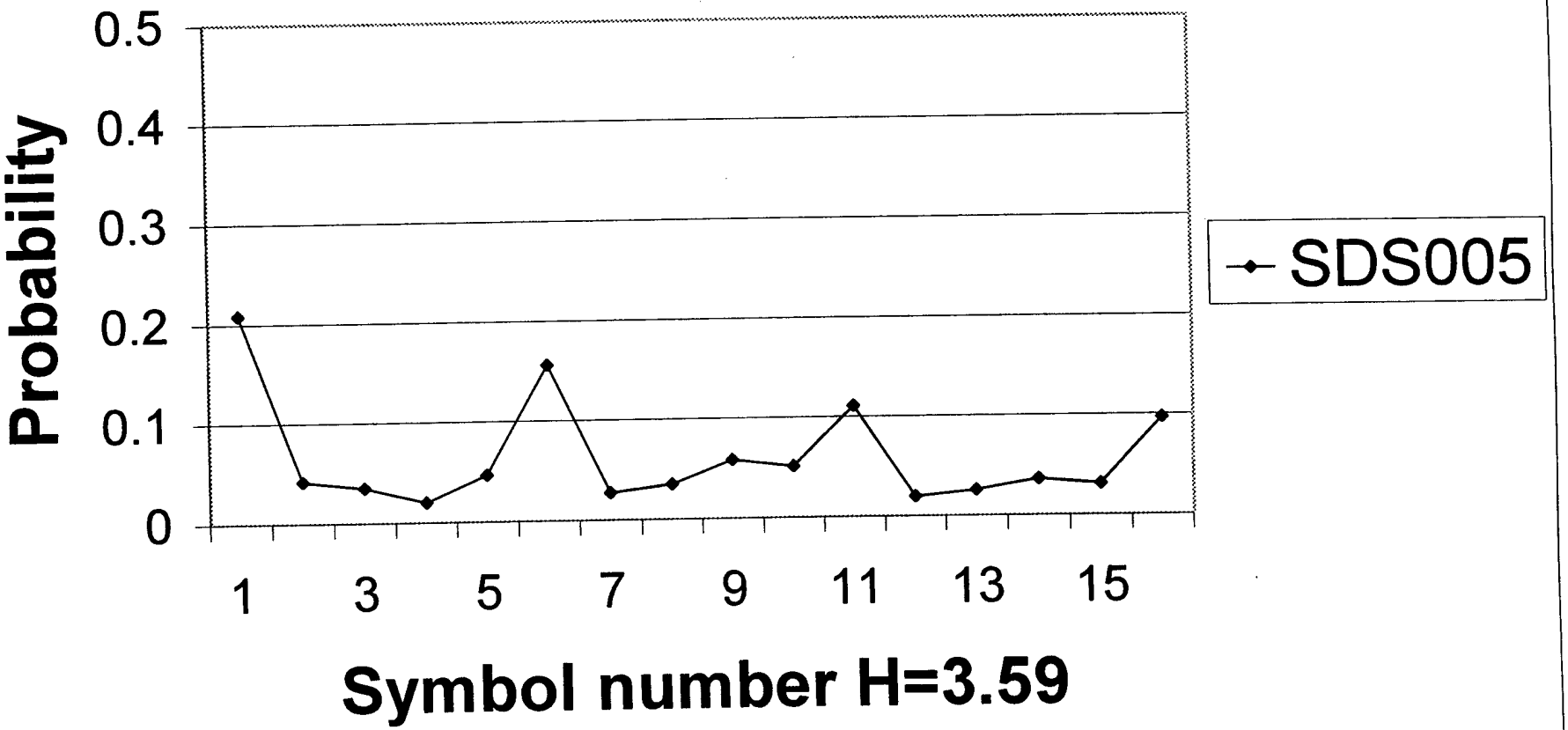

Frequency information for one symbol of SDS002
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS002
Symbol Number H=3.3

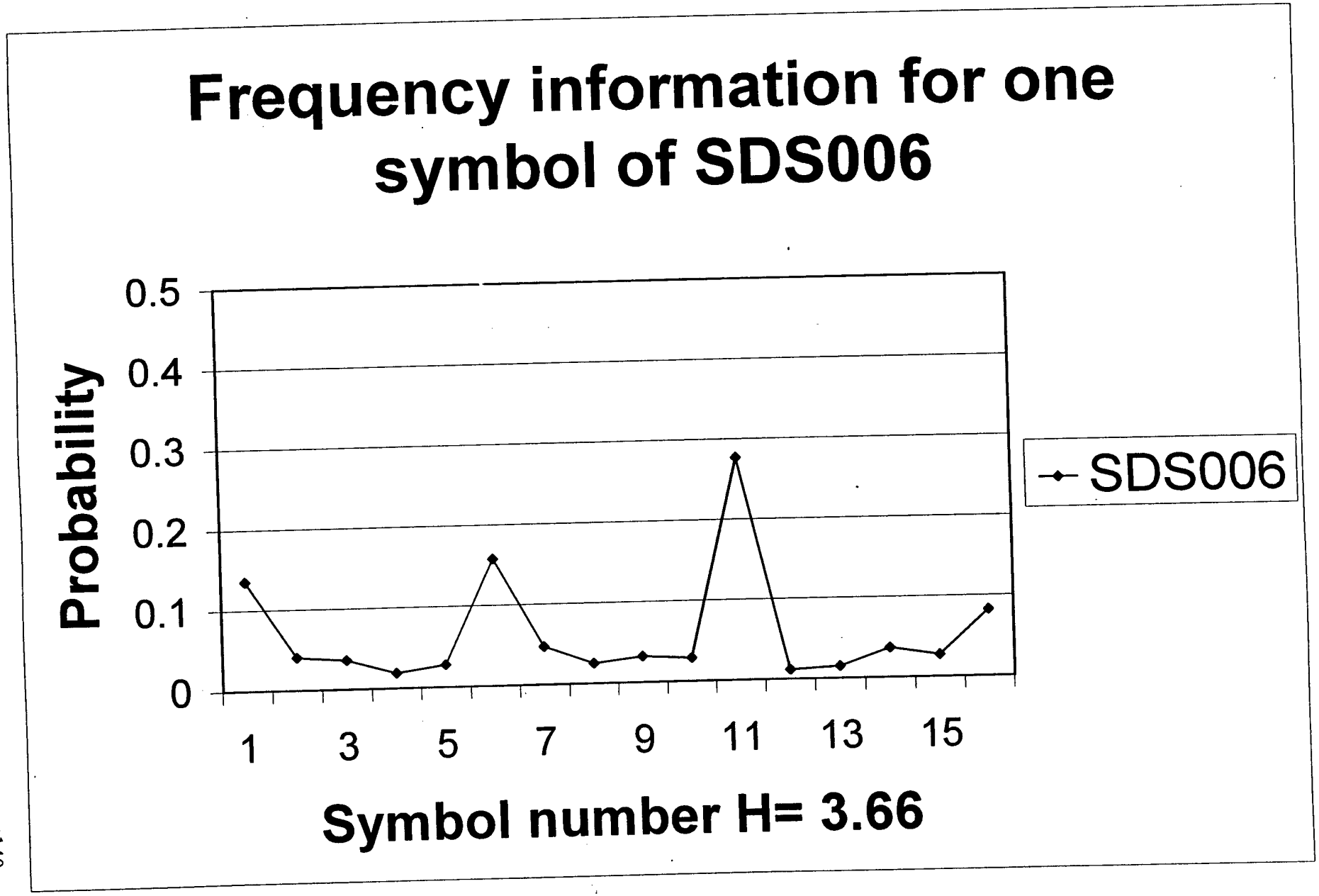
Frequency information for one symbol of SDS003



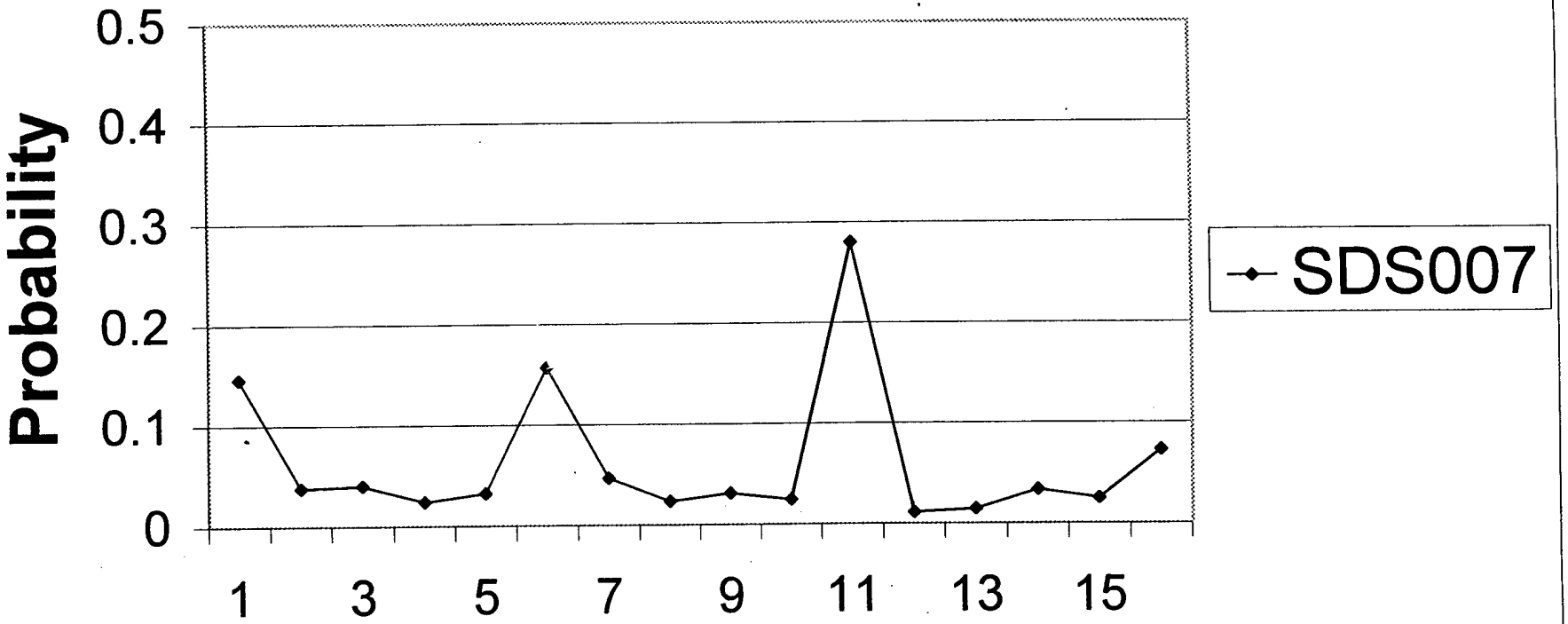

Frequency information for one symbol of SDS004
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS004
Symbol number H= 3.36

Frequency information for one symbol of SDS005

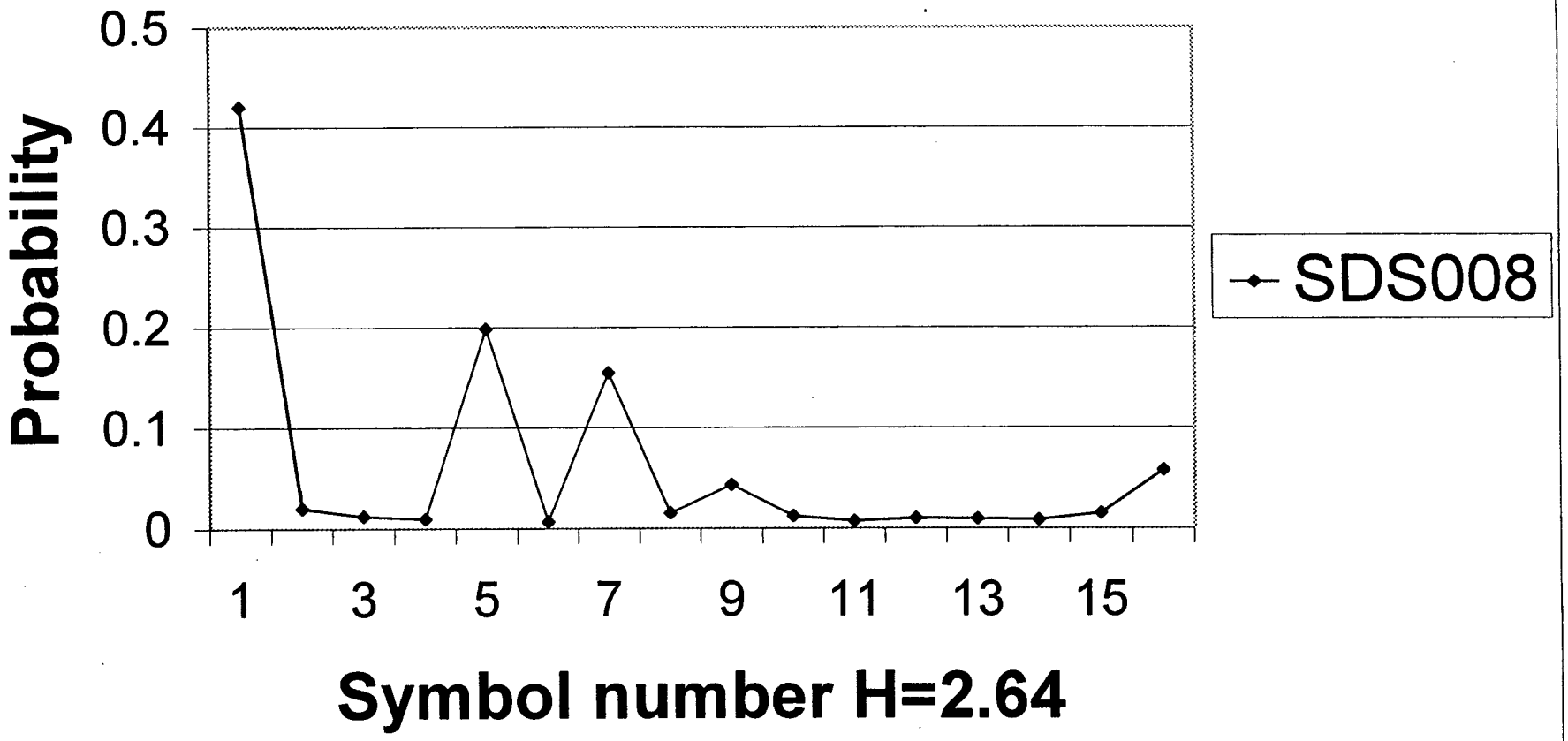


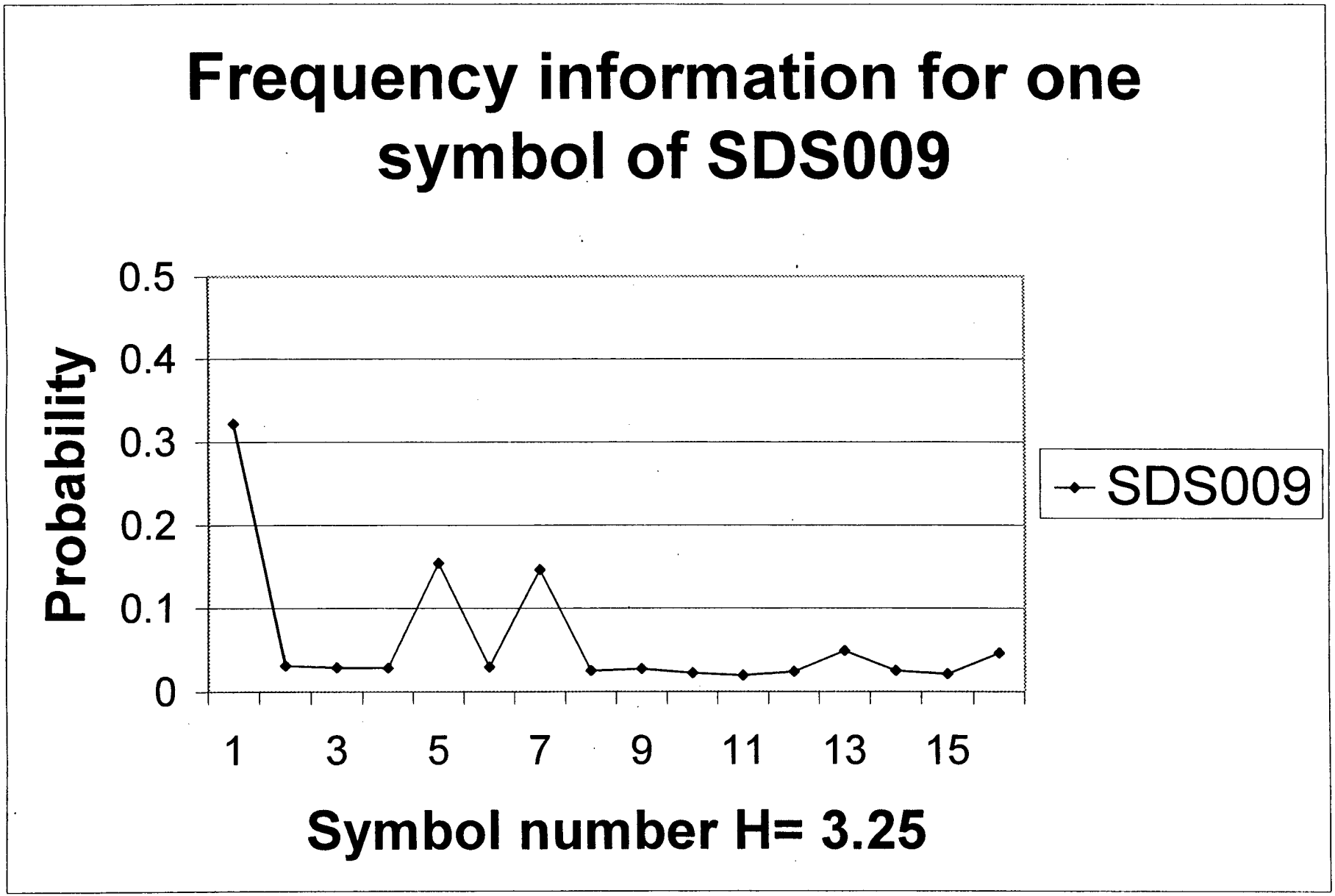

Frequency information for one symbol of SDS006
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS006
Symbol number H= 3.66

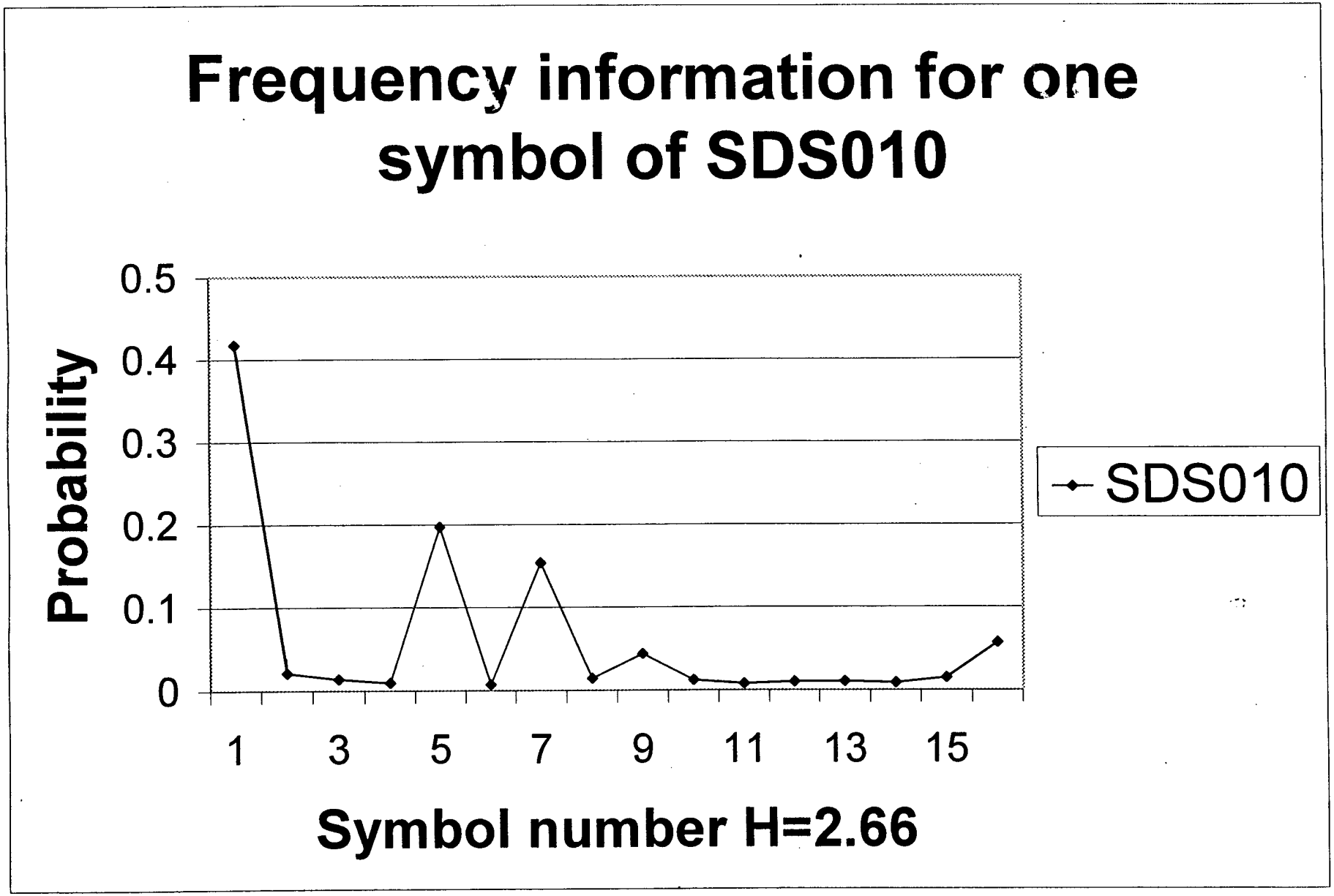
Frequency information for one symbol of SDS007

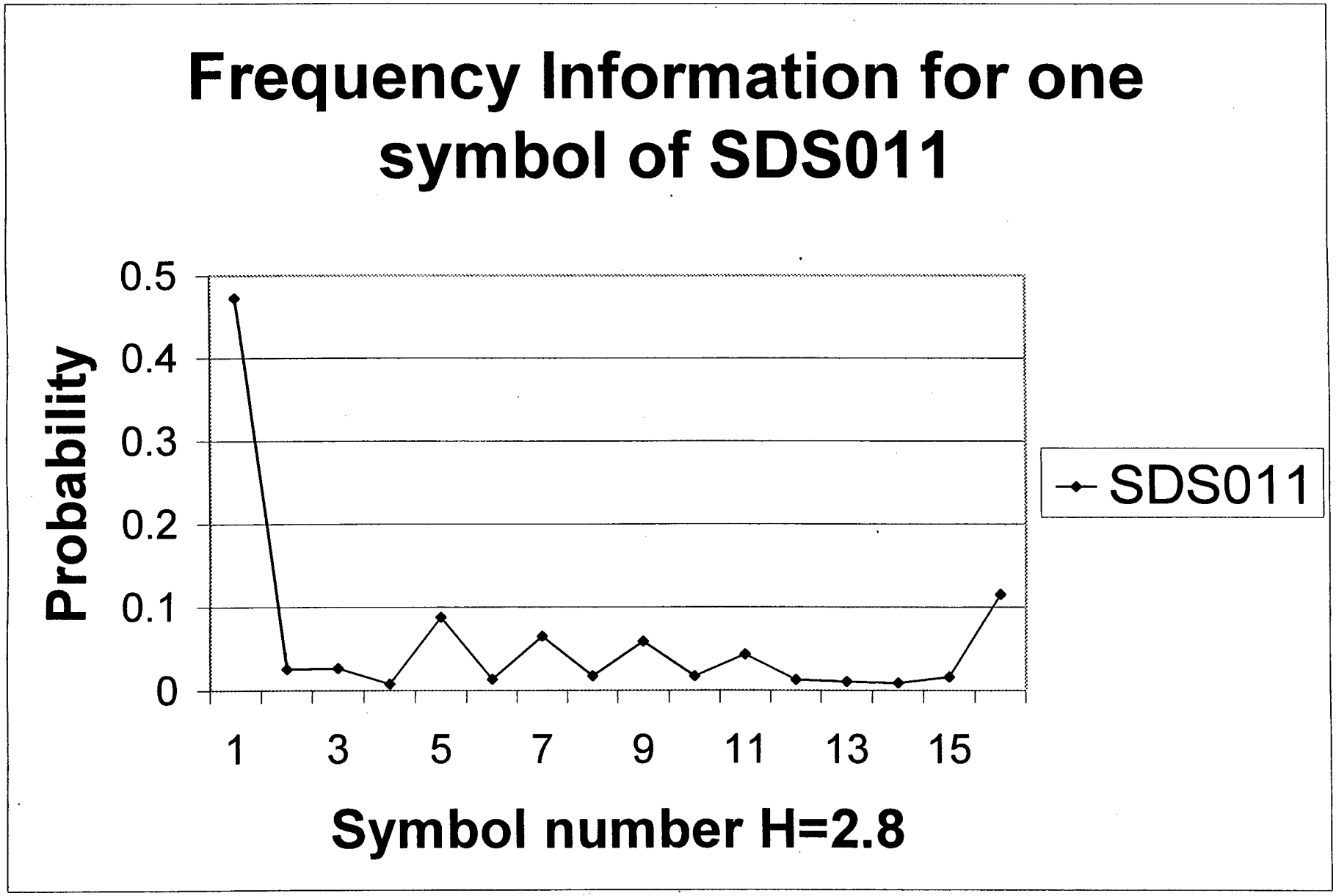


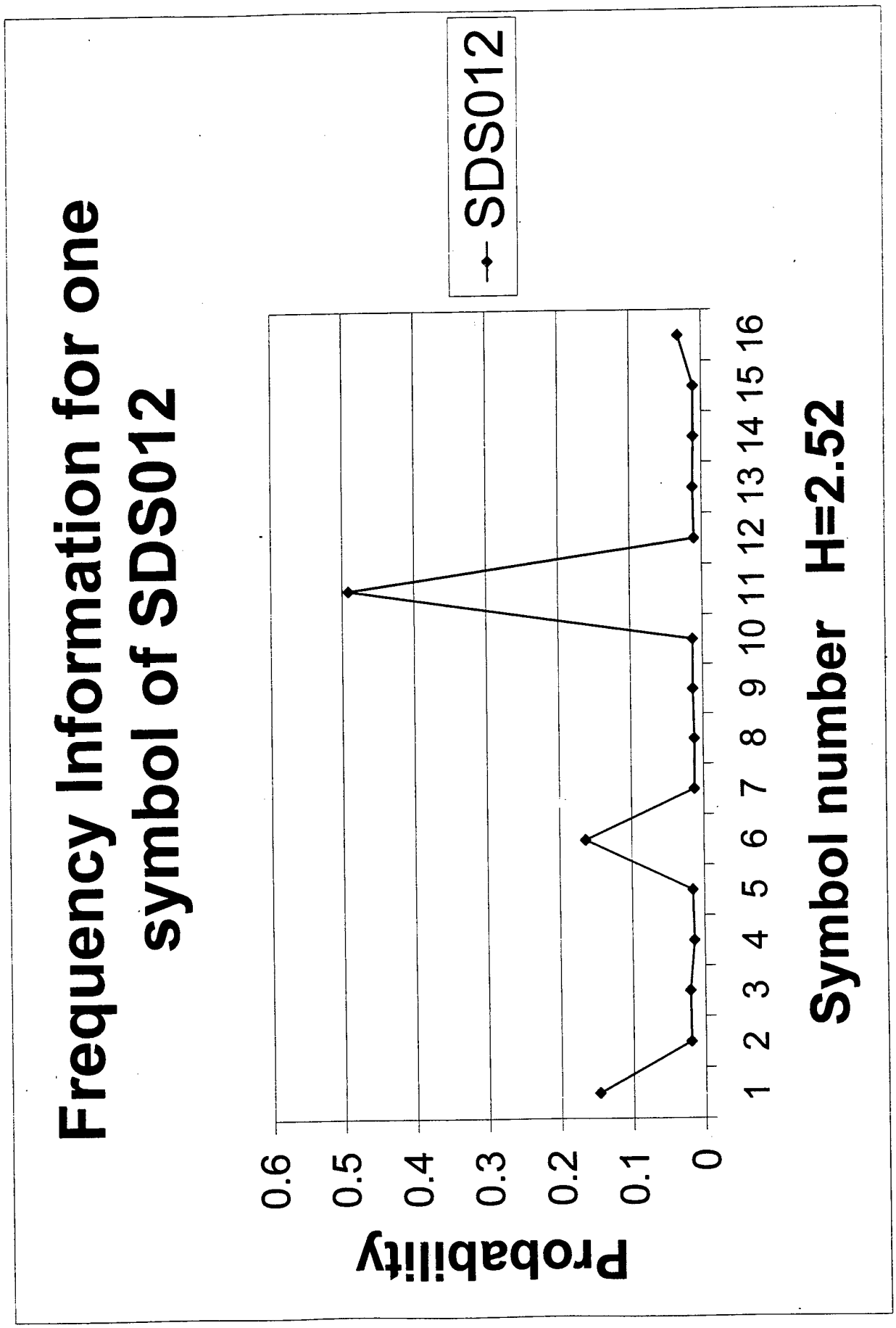
Frequency information for one symbol of SDS008

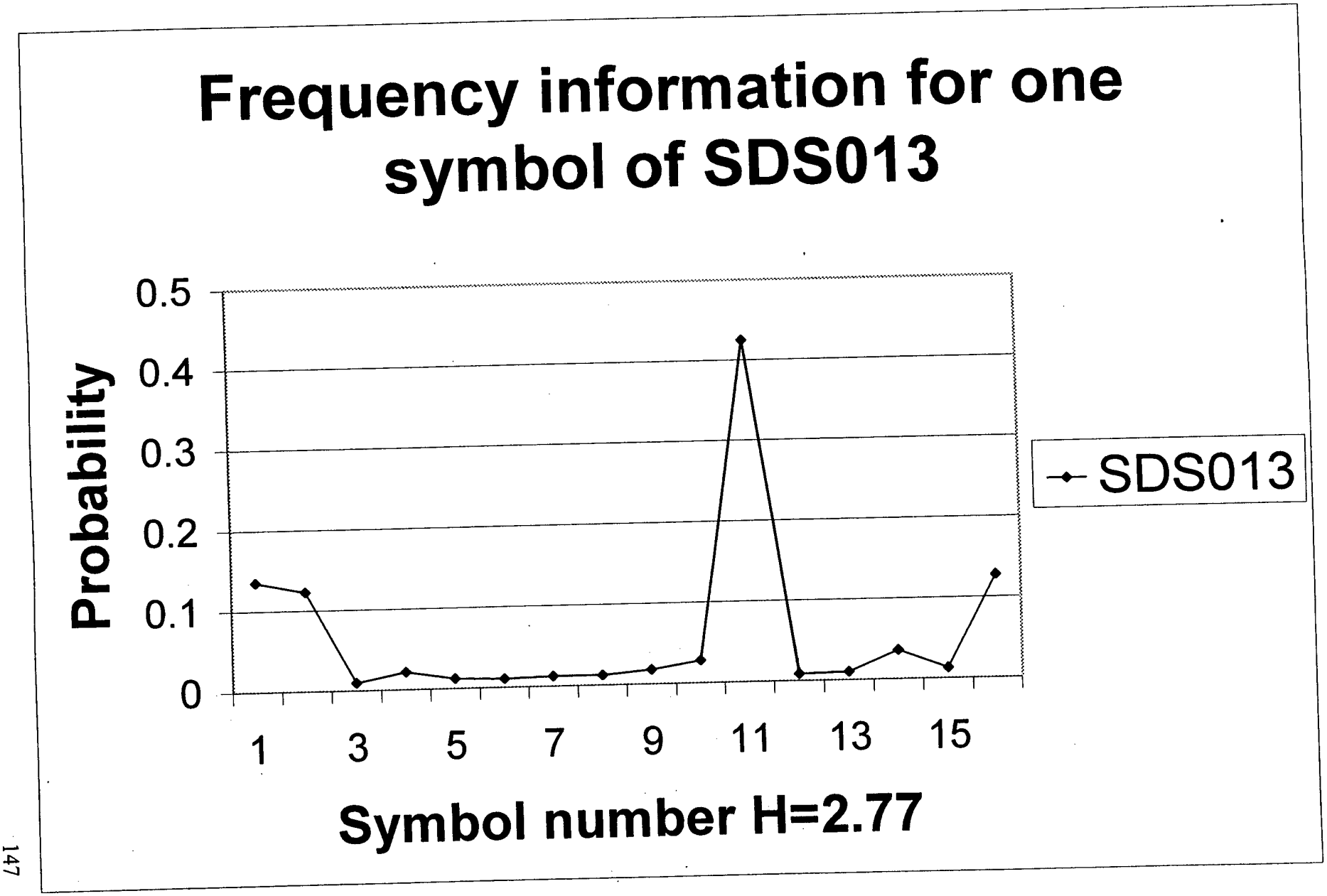



Frequency information for one symbol of SDS009
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS009
Symbol number H= 3.25

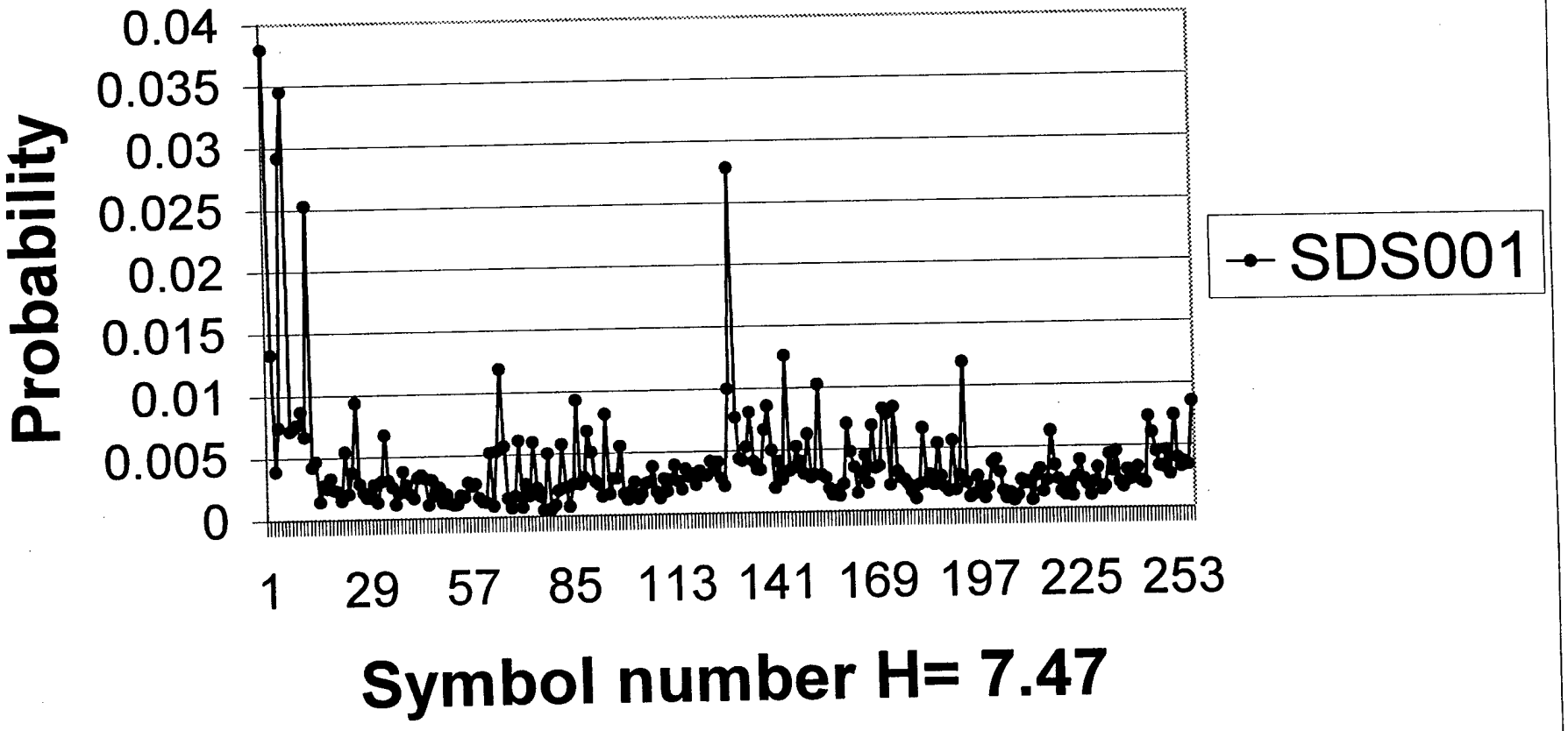

Frequency information for one symbol of SDS010
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS010
Symbol number H=2.66

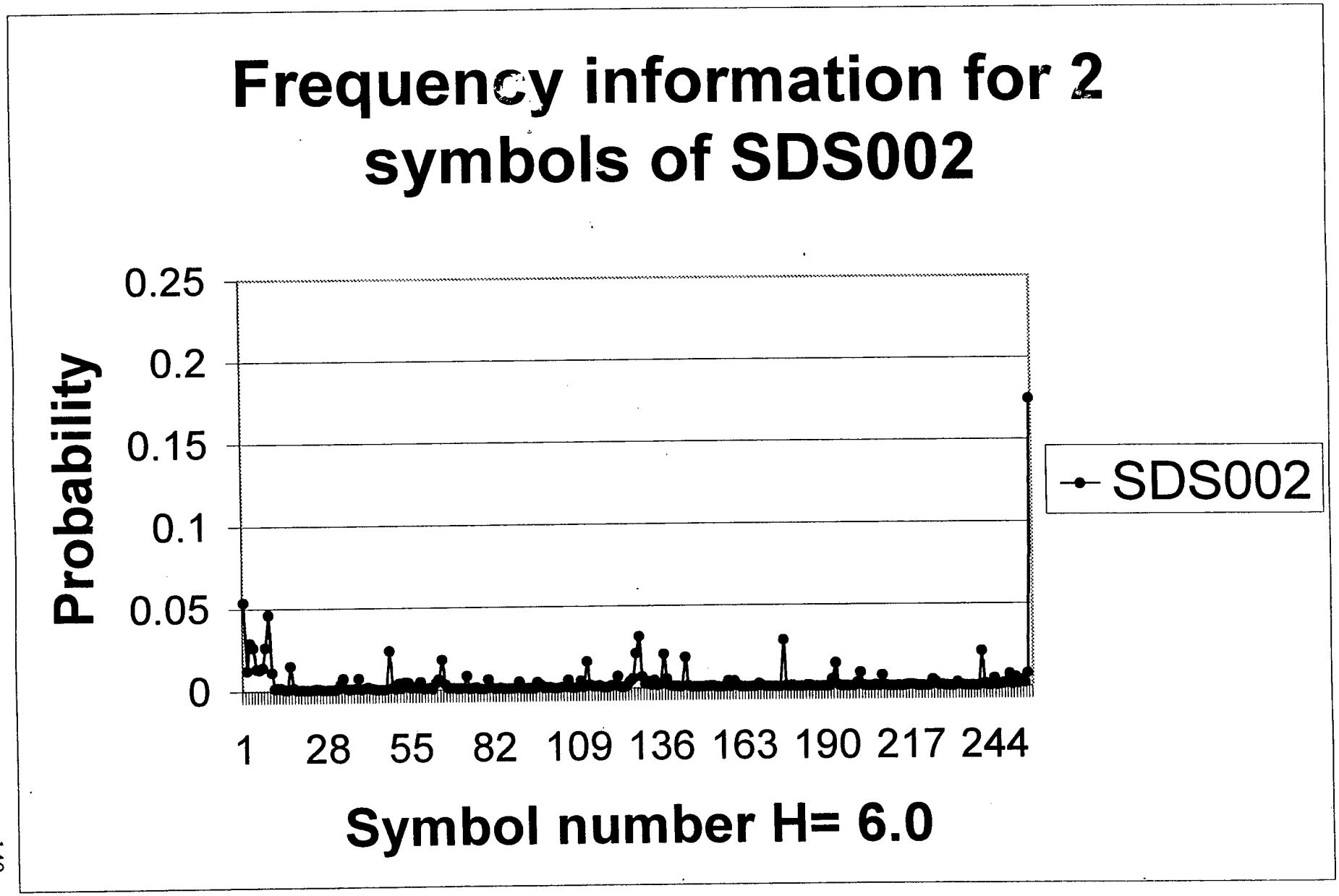

Frequency Information for one symbol of SDS011
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS011
Symbol number H=2.8

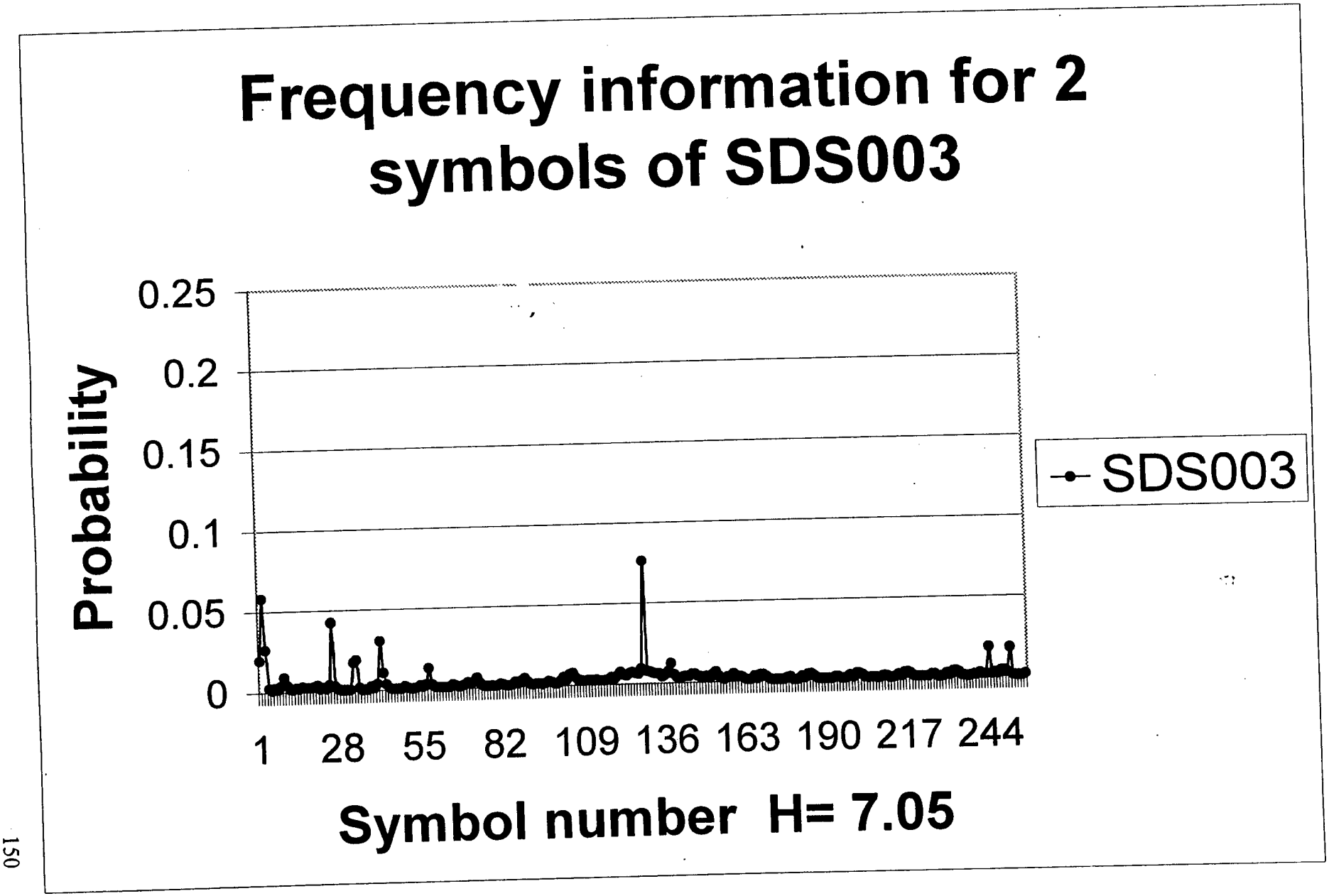

Frequency Information for one symbol of SDS012
SDS012
Probability
0.6
0.5
0.4
0.3
0.2
0.1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Symbol number H=2.52

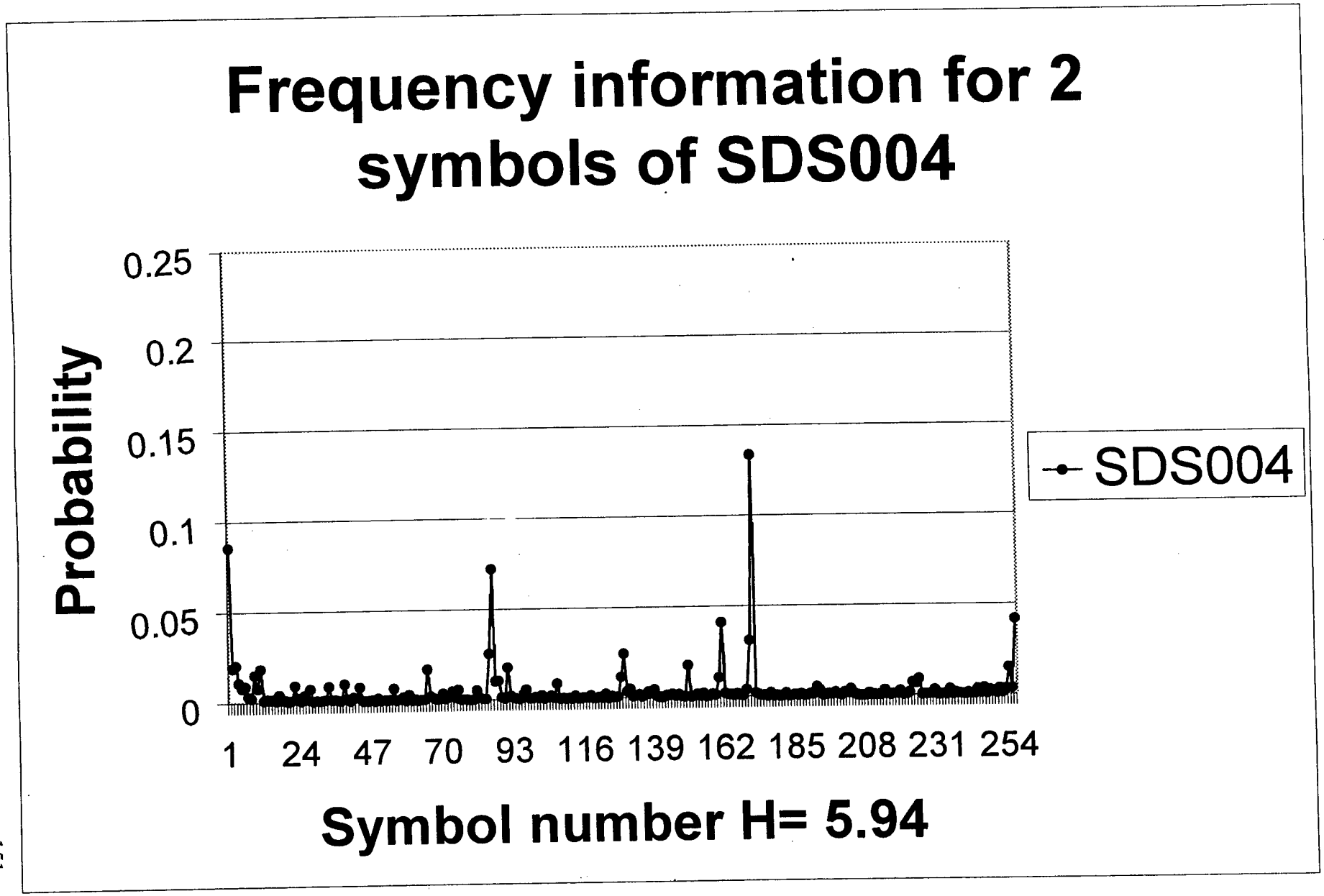

Frequency information for one symbol of SDS013
Probability
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
SDS013
Symbol number H=2.77

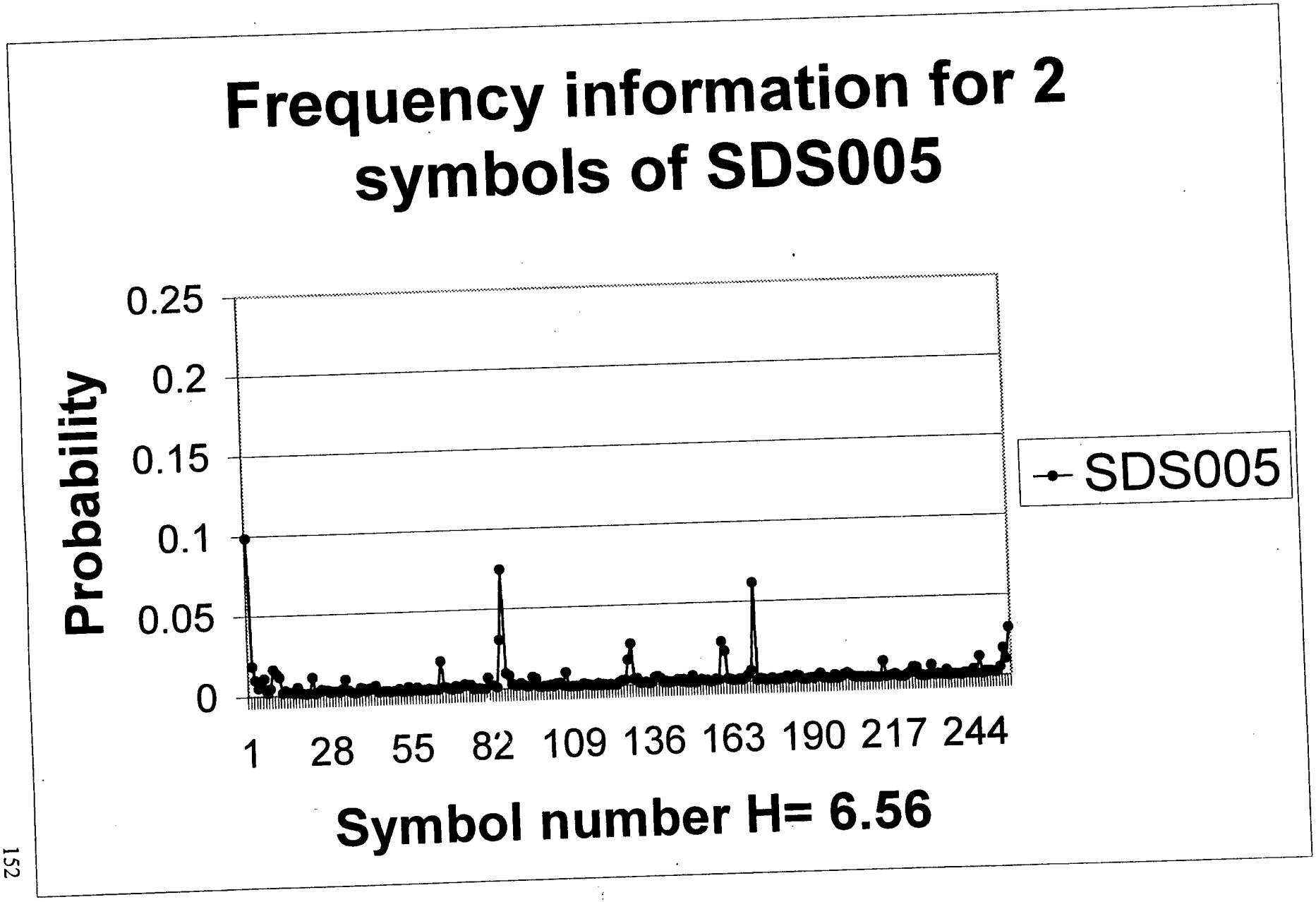
Frequency information for two symbols of SDS001



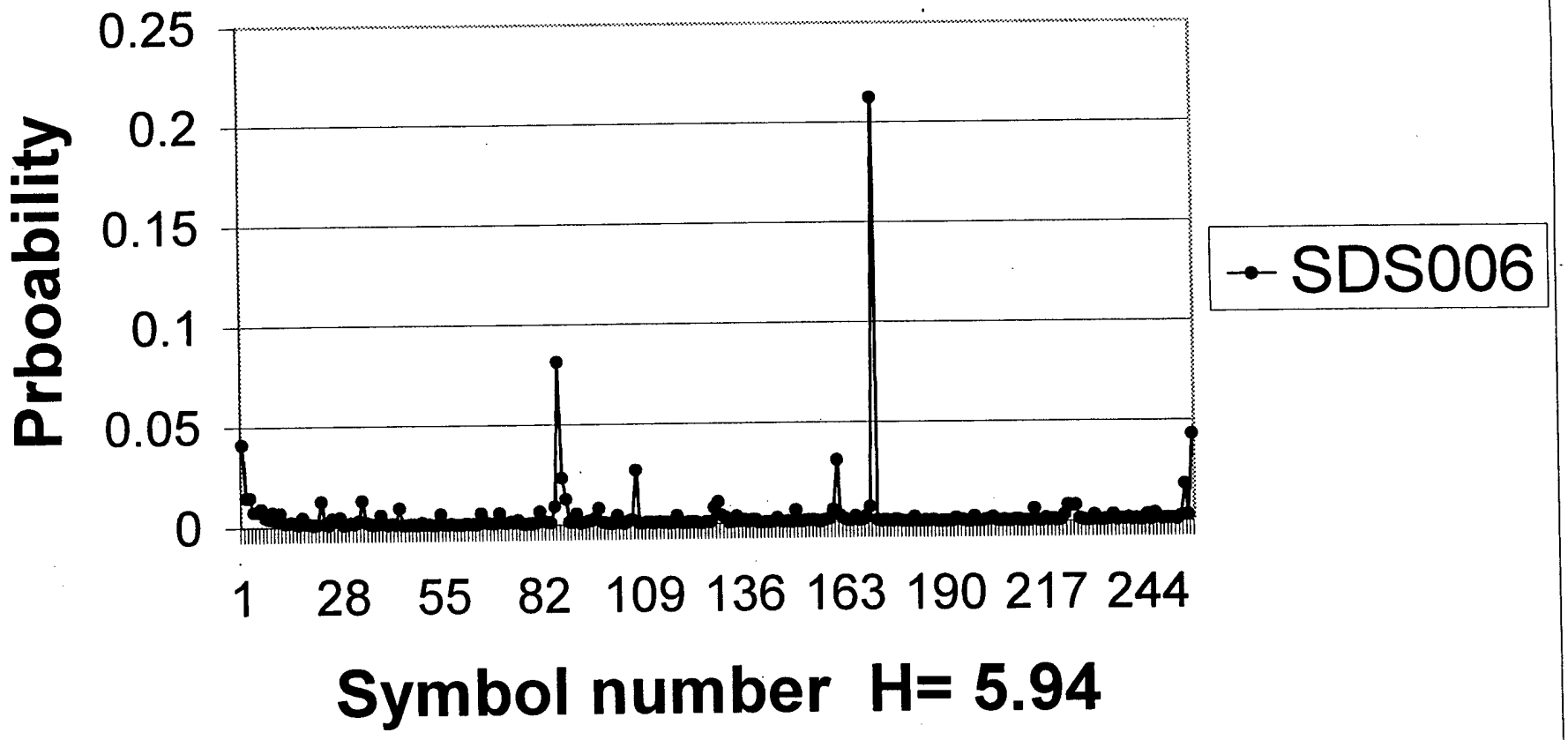

Frequency information for 2 symbols of SDS002
Probability
0.25
0.2
0.15
0.1
0.05
0
1
28
55
82
109
136
163
190
217
244
Symbol number H= 6.0
SDS002

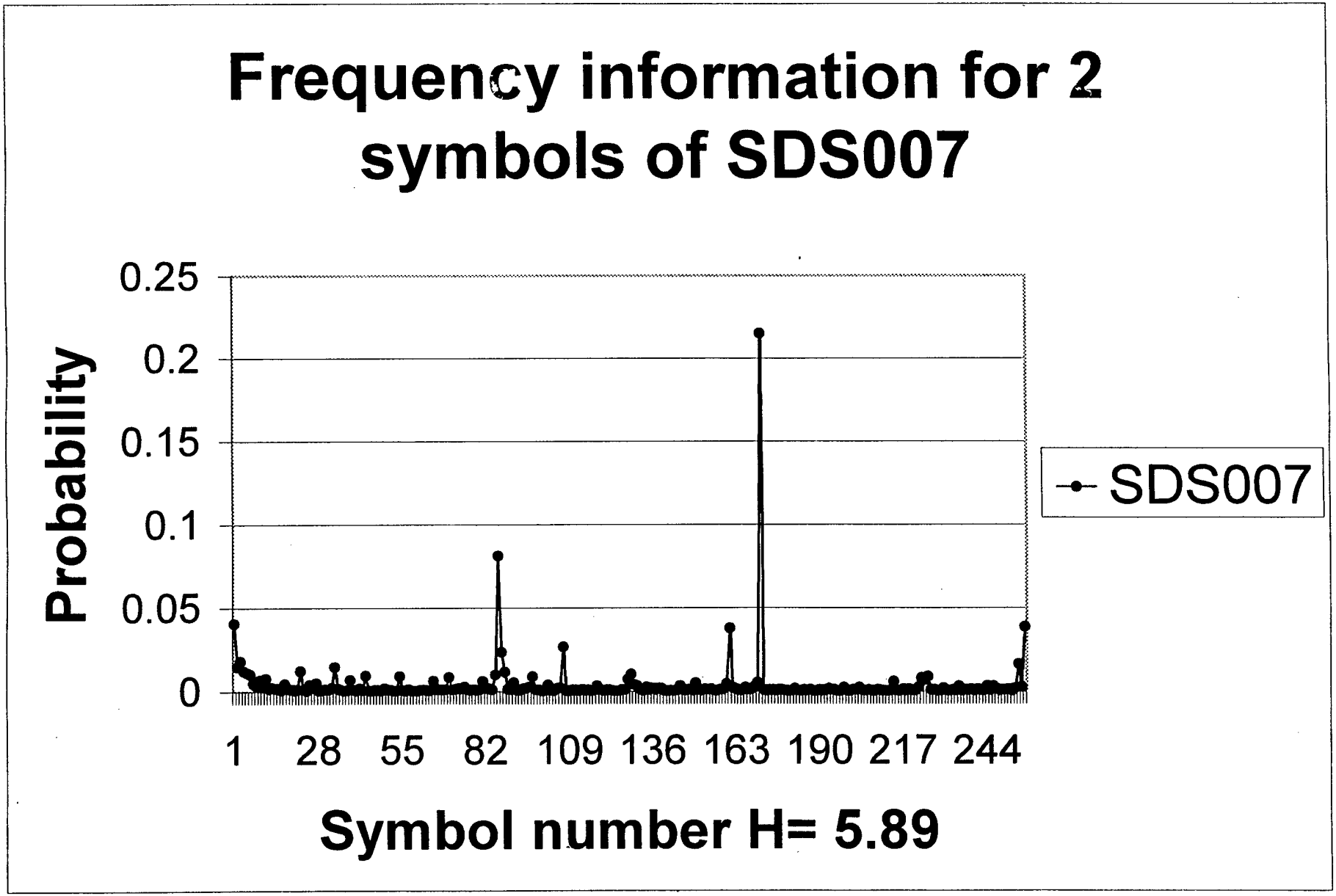

Frequency information for 2 symbols of SDS003
Probability
0.25
0.2
0.15
0.1
0.05
0
1
28
55
82
109
136
163
190
217
244
Symbol number H= 7.05
SDS003

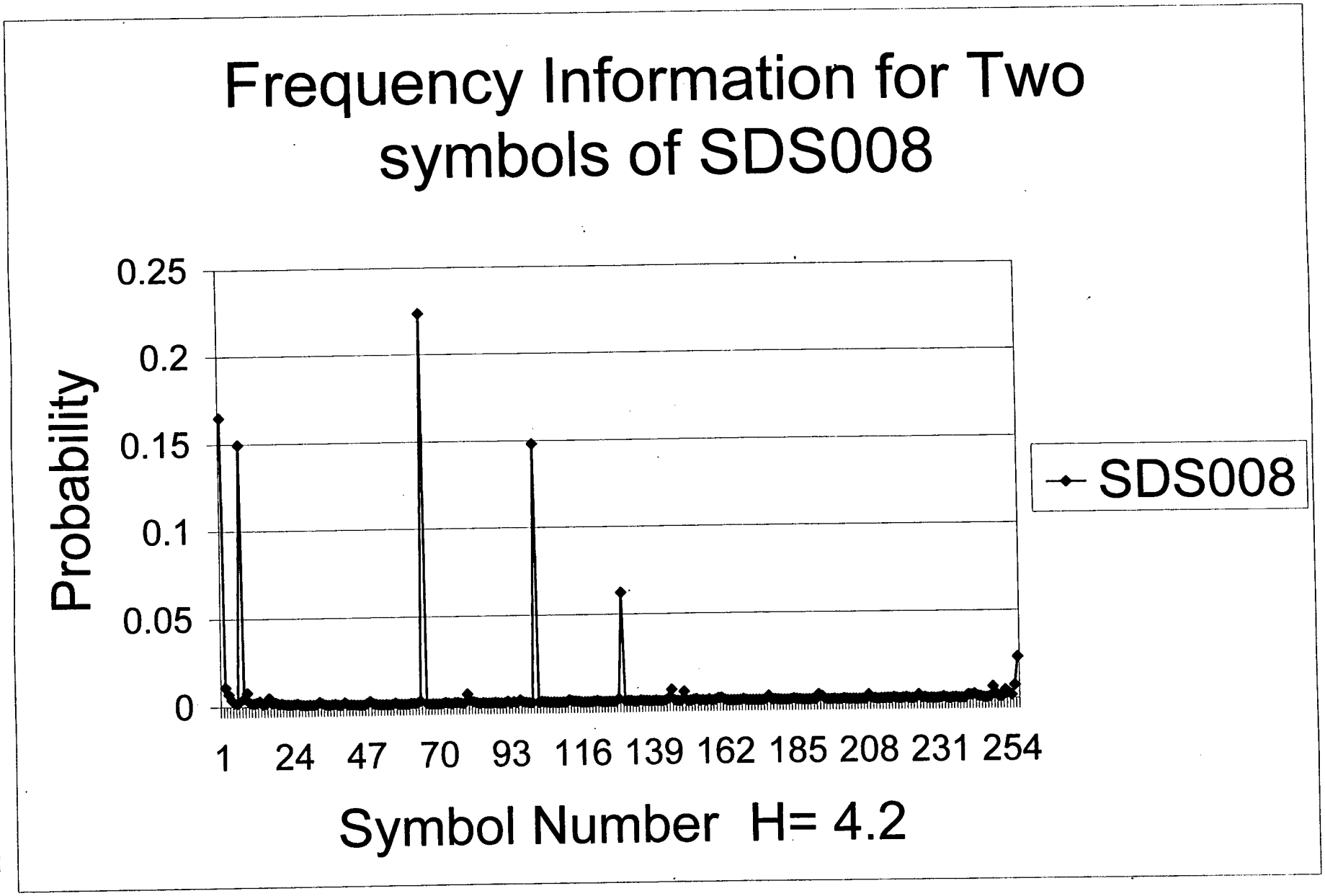

Frequency information for 2 symbols of SDS004
Probability
0.25
0.2
0.15
0.1
0.05
0
1
24
47
70
93
116
139
162
185
208
231
254
SDS004
Symbol number H= 5.94

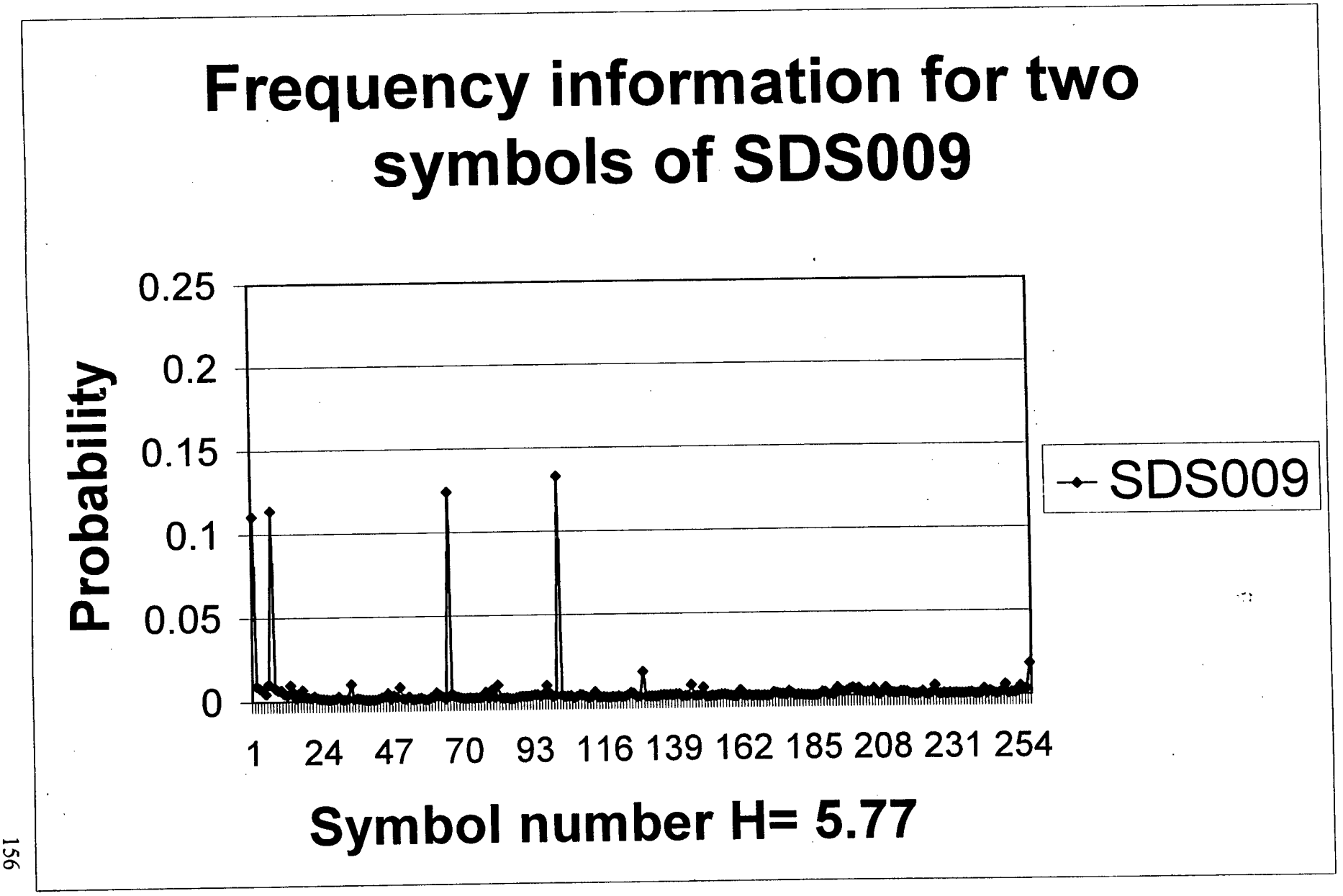

Frequency information for 2 symbols of SDS005
Probability
0.25
0.2
0.15
0.1
0.05
0
1
28
55
82
109
136
163
190
217
244
SDS005
Symbol number H= 6.56

Frequency information for 2 symbols of SDS006

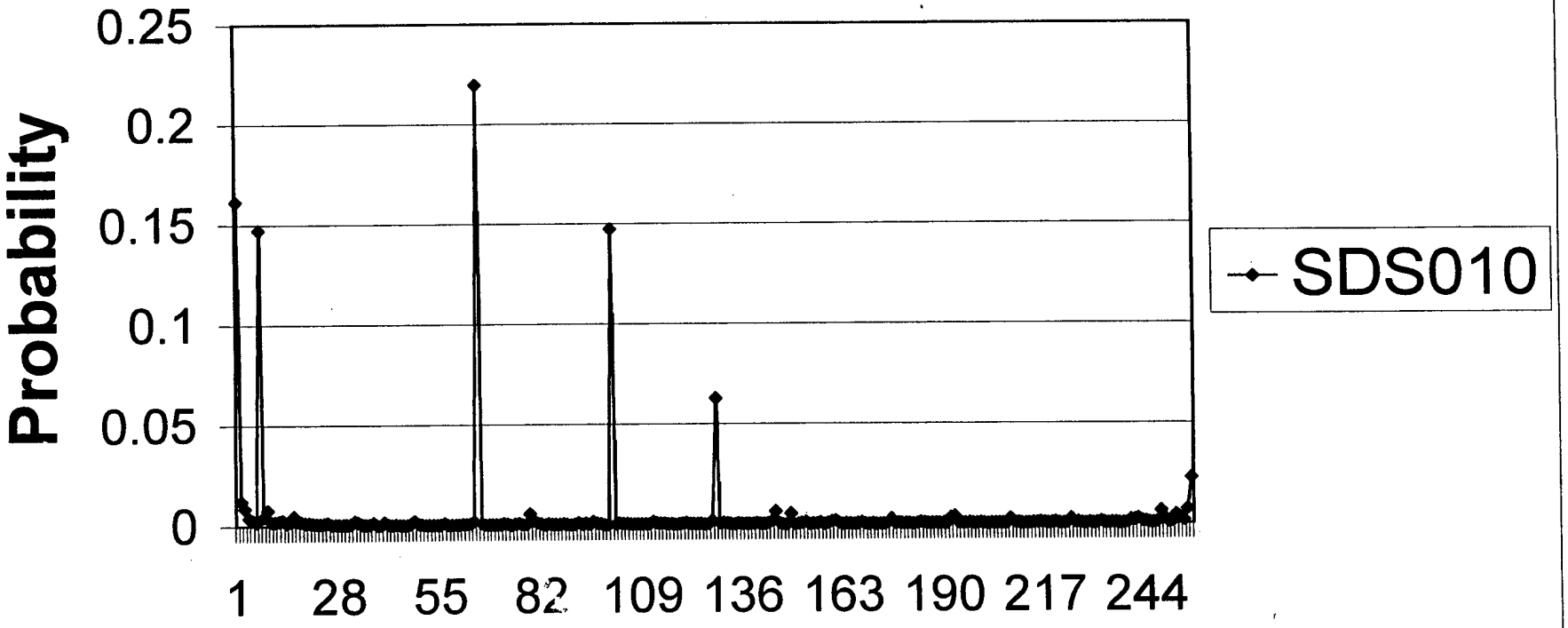


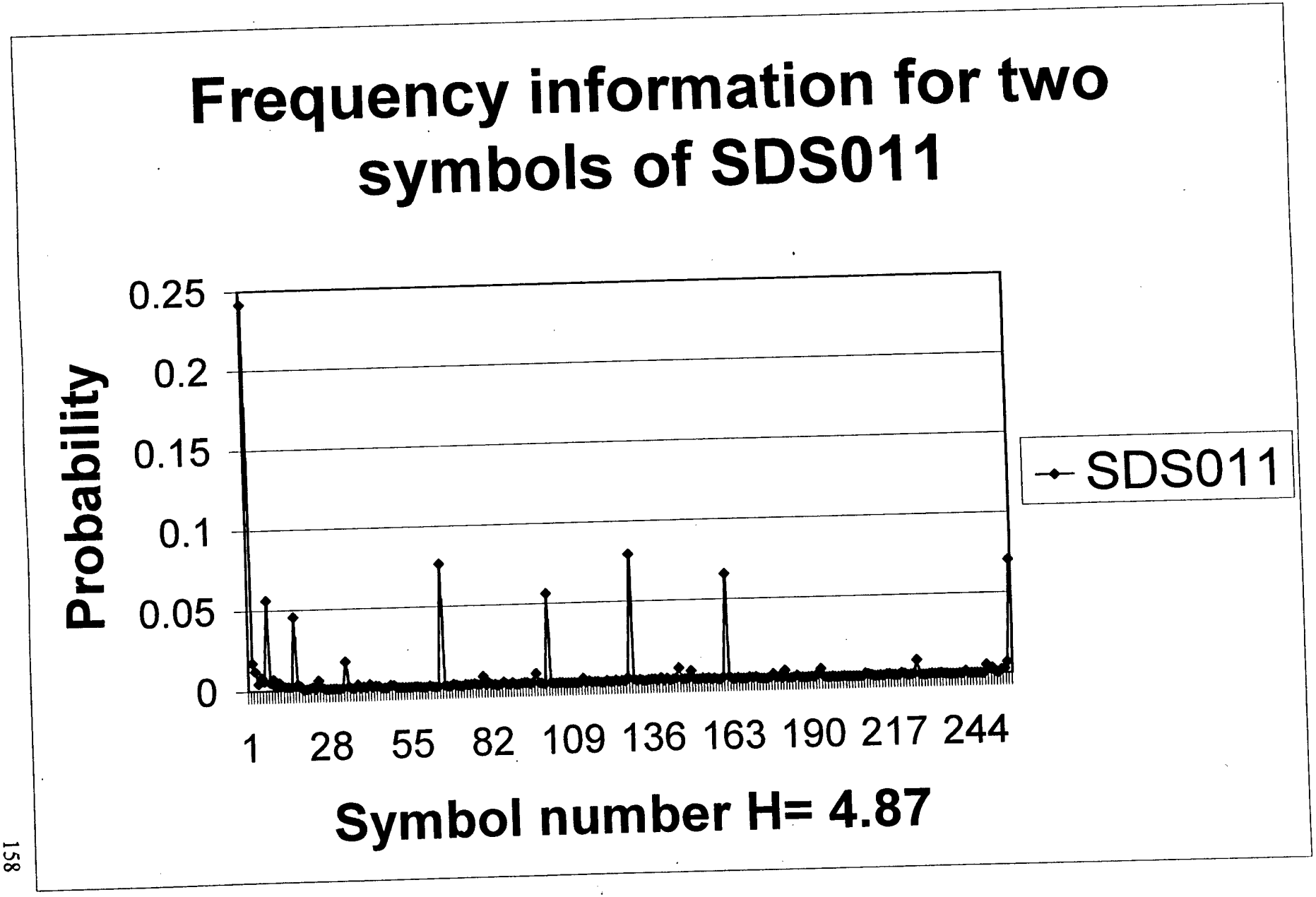

Frequency information for 2 symbols of SDS007
Probability
0.25
0.2
0.15
0.1
0.05
0
1
28
55
82
109
136
163
190
217
244
SDS007
Symbol number H= 5.89

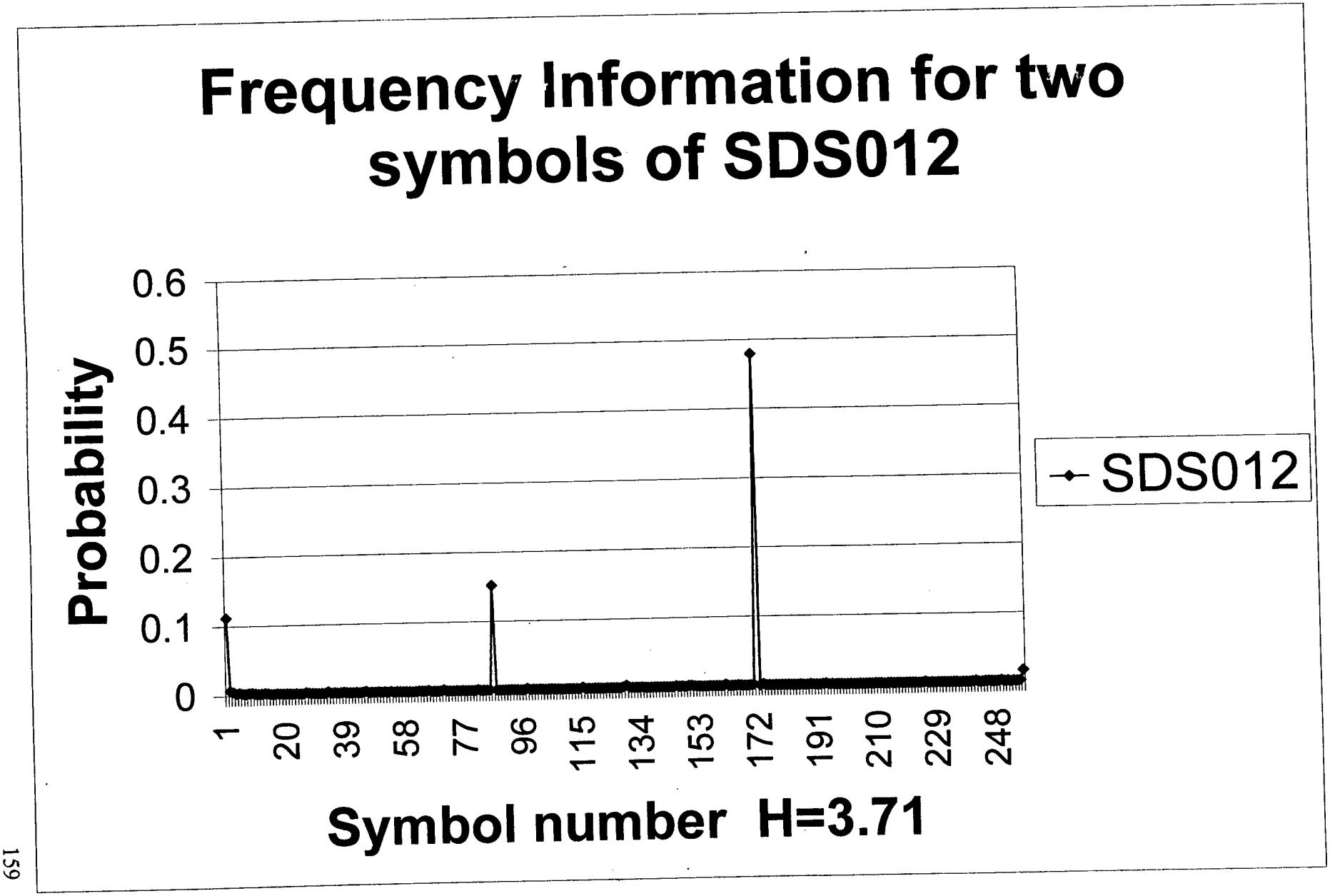
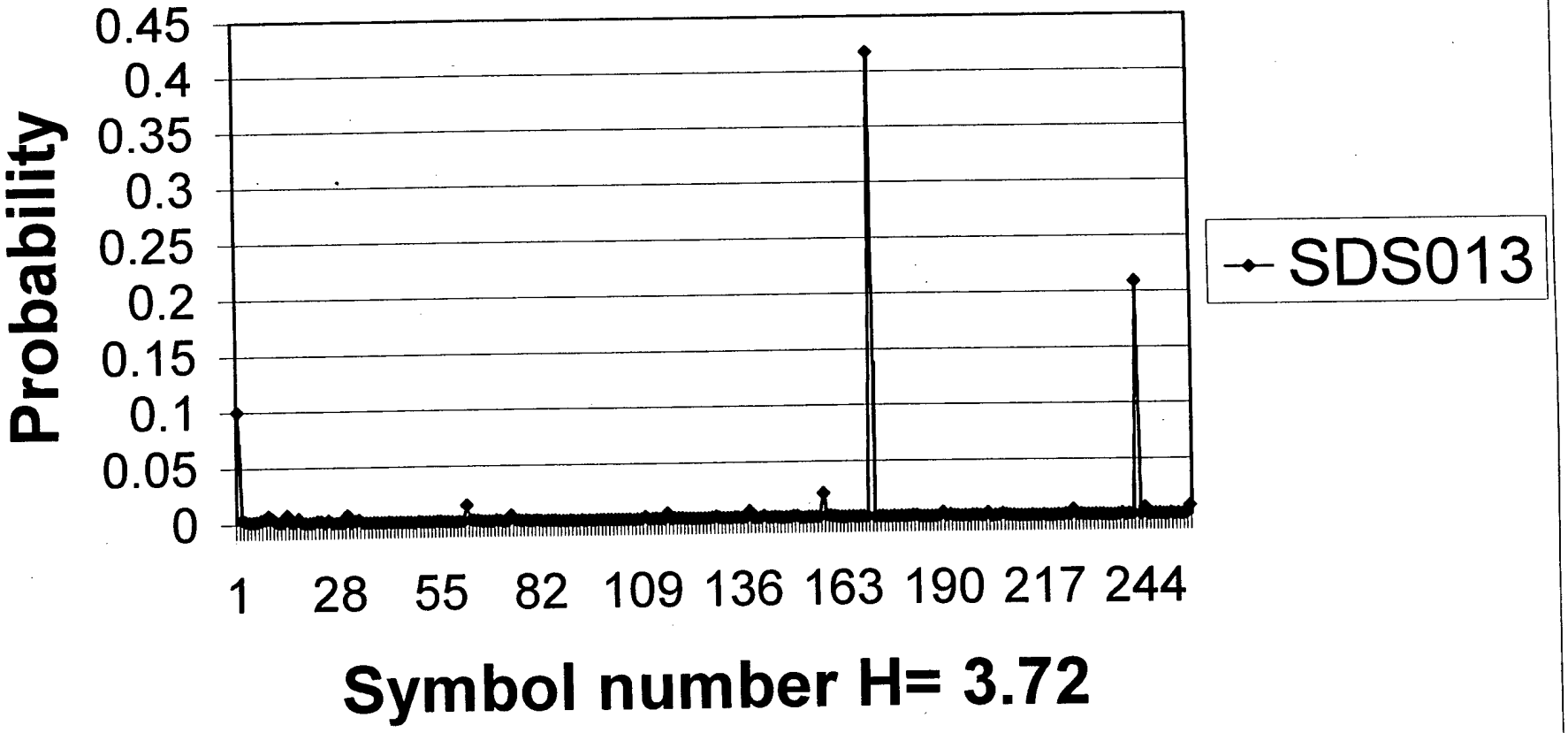

Frequency Information for Two symbols of SDS008
Probability
0.25
0.2
0.15
0.1
0.05
0
1
24
47
70
93
116
139
162
185
208
231
254
Symbol Number H= 4.2
SDS008


Frequency information for two symbols of SDS009
Probability
0.25
0.2
0.15
0.1
0.05
0
1
24
47
70
93
116
139
162
185
208
231
254
SDS009
Symbol number H= 5.77

Frequency information for two symbols of SDS010




Frequency information for two symbols of SDS011
Probability
0.25
0.2
0.15
0.1
0.05
0
1
28
55
82
109
136
163
190
217
244
SDS011
Symbol number H= 4.87


Frequency Information for two symbols of SDS012
Probability
0.6
0.5
0.4
0.3
0.2
0.1
0
1
20
39
58
77
96
115
134
153
172
191
210
229
248
Symbol number H=3.71
SDS012

Frequency information for two symbols of SDS013